S+ArrayAnalyzer 2.0 User's Guide

June 2004

Insightful Corporation
Seattle, Washington

Proprietary Notice

Insightful Corporation owns the S+ARRAYANALYZER software program, and its documentation. Both the program and documentation are copyrighted with all rights reserved by Insightful Corporation.

S+ARRAYANALYZER provides access to the Bioconductor R packages for microarray analysis, which are free software. The affy, annotate, Biobase, edd, gcrma, genefilter, geneplotter, LPEtest, marrayClasses, marrayInput, marrayNorm, marrayPlots, matchprobes, multtest, ROC, and vsn libraries are copyrighted © 2004 by Insightful Corporation. These libraries are free software that are redistributed and modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; version 2.1 of the License. The S+ARRAYANALYZER software is covered by a separate license agreement.

The correct bibliographical reference for this document is as follows:

S+ARRAYANALYZER 2.0 User's Guide, Insightful Corporation, Seattle, WA.

Printed in the United States.

Copyright Notice

Copyright © 1987-2004, Insightful Corporation. All rights reserved.

Insightful Corporation
1700 Westlake Avenue N, Suite 500
Seattle, WA 98109-3044
USA

Trademarks

Insightful, Insightful Corporation, "Insightful intelligence from data," S-PLUS, S+, S-PLUS Graphlets, Graphlets, and InFact are registered trademarks of Insightful Corporation. ArrayAnalyzer, FinMetrics, NuOpt, SeqTrial, Wavelets, and SpatialStats are trademarks of Insightful Corporation. All product names mentioned herein may be trademarks or registered trademarks of their respective companies.

ACKNOWLEDGMENTS

Insightful S+ARRAYANALYZER uses Bioconductor packages that represent state-of-the-art work from a collection of leading statisticians. Insightful would like to recognize these contributors:

affy - Rafael A. Irizarry, Laurent Gautier, and Leslie M. Cope

AnnBuilder - Jianhua Zhang

annotate - Robert Gentleman

Biobase - Robert Gentleman and Vincent Carey

edd - Vincent Carey

genefilter - Robert Gentleman and Vincent Carey

geneplotter - Robert Gentleman

marrayNorm - Sandrine Dudoit, Yee Hwa (Jean) Yang

marrayClasses - Sandrine Dudoit, Yee Hwa (Jean) Yang

marrayInput - Sandrine Dudoit, Yee Hwa (Jean) Yang

marrayPlots - Sandrine Dudoit, Yee Hwa (Jean) Yang

multtest - Yongchao Ge, Sandrine Dudoit

rhdh5 - Byron Ellis, Robert Gentleman

ROC - Vincent Carey

CONTENTS

Acknowledgments iii

Chapter 1 Introduction To Microarray Data 1

- **Welcome!** 2
- **Supported Platforms and System Requirements** 6
- **Genomics and Differential Expression** 9
- **Microarray Data** 11

Chapter 2 Examples: Affymetrix MAS Data 15

- **Affymetrix Data Analysis Workflow** 16
- **Experimental Design** 17
- **One-Way Design** 18
- **From the Command Line** 54
- **References** 65

Chapter 3 Examples: Affymetrix Probe-Level Data 67

- **Affymetrix Probe-Level Data Analysis Workflow** 68
- **Two-Sample Design** 69
- **Two-Way Design** 93
- **References** 123

Chapter 4 Examples: Two-Color Data 125

- **Two-Color Data Analysis Workflow** 126

Two-Sample Design 127
Two-Way Reference Design 147
From the Command Line 174

Chapter 5 Quality Control Diagnostics and Filtering 191
Quality Control Diagnostics 192
Filtering 203

Chapter 6 Pre-Processing and Normalization 207
Introduction 209
Normalization 210
Ideas in Normalization 214
Diagnostic Plots 218
Normalization Methods for Two-Channel Data 220
Pre-Processing and Normalization for Affymetrix Probe-Level Data 232
Normalization Methods for Affymetrix MAS Data 251
References 256

Chapter 7 Differential Expression Testing 259
Introduction 260
Statistical Tests 261
Controlling Type I Error Rates 265
GUI for Two Sample Testing 270
GUI for LPE Testing 276
GUI for ANOVA Testing 280
Differential Expression Analysis Plots 286
Differential Expression Summary Table Output 294
References 298

Chapter 8 Cluster Analysis **301**

Introduction 302

Hierarchical Methods 304

Partitioning Methods 308

Examples from the GUI 310

Examples from the Command Line 320

References 329

Chapter 9 Annotation and Gene List Management **331**

Annotation and Gene List Management Functionality 332

Annotation Libraries 334

Appendix A: Creating a Design File **369**

Introduction 370

Format Specification 371

Appendix B: Importing data **385**

Introduction 386

Index **389**

INTRODUCTION TO MICROARRAY DATA

1

Welcome! **2**
- Features 3
- Goals 4
- Libraries 4

Supported Platforms and System Requirements **6**
- Installing and Running S+ARRAYANALYZER 6
- Online Help 6
- Online Reference 7
- Technical Support 7

Genomics and Differential Expression **9**

Microarray Data **11**
- Affymetrix Arrays 13
- Two-Color Arrays 14

WELCOME!

S+ARRAYANALYZER is an S-PLUS module that provides you with a powerful tool for analyzing Affymetrix MAS 5, CHP, and CEL data, and two-channel microarray data. Using either the graphical user interface (GUI) or the **Commands** window, you can perform statistical analysis to determine differential gene expression in microarrays, fundamental to the rapidly growing field of functional genomics. In S+ARRAYANALYZER, you can access functions in a

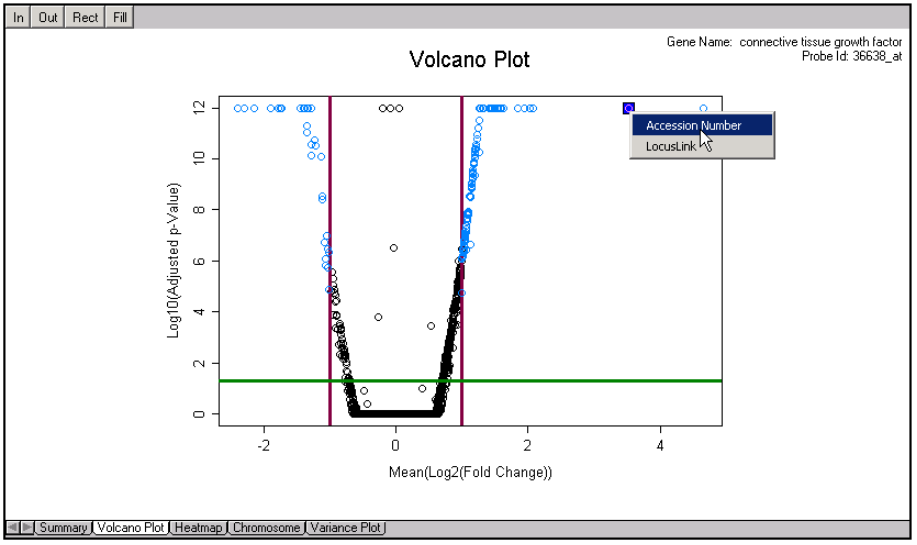


Figure 1.1: *Sample volcano plot with Affymetrix CEL data, generated by the* ***(Differential Expression Analysis) LPE Test*** *dialog in S+ARRAYANALYZER. This plot shows genes that would be false positives and false negatives based on the fold-change criteria alone, but you can also quickly find significantly differentially expressed genes after adjustment for FWER/FDR (family-wise error rate/family detection rate). With the annotation turned on in an HTML display, you can click specific genes to determine which are in the significant zone.*

collection of libraries based on the Bioconductor project, an open source and open development software project developed by leading statisticians with the goal of providing tools for current microarray and genomics research. S+ARRAYANALYZER 2.0 is based on Bioconductor version 1.3, with some libraries ported from 1.4.

Features

The S+ARRAYANALYZER module helps you analyze microarray data using these built-in features:

- **Enhanced microarray data import**: Supported file types now include Affymetrix MAS 5 (ASCII or Excel), CHP (binary), and CEL (ASCII or binary), and any two-channel microarray data (ASCII or Excel).
- **Improved pre-processing**: Updated Affymetrix probe level methods (e.g., affy, gcrma, matchprobes from the Bioconductor libraries), and added QC diagnostics (PCA, RNA degradation) and filtering (genes, arrays).
- **Added cluster analysis methods**: A palette of supervised and unsupervised learning methods (e.g., `mclust`) are included in this release.
- **Added annotation for derived genelists**: For example, you can import Affymetrix NetAffx annotation data, and your customers can update the data from the Affymetrix Web site. You can also connect to online databases for genelist annotation analysis, and integrate with OntoExpress and DAVID/EASE.
- **New gene list management**: Allows you to merge gene lists from testing different contrasts in an analysis.
- **Enhanced data normalization methods**: Added vsn and quantiles to Affymetrix summary data, vsn to Affymetrix CEL, and a between-array quantile normalization method to two-color experiments.
- **Expanded differential expression testing**: This uses the LPEtest and multtest libraries, and 1-way and 2-way ANOVA.
- **New quality control diagnostics**: Provides an assortment of graphical tools for assessing the quality of your experimental data.
- **Enhanced S-PLUS Graphlets®**: For annotation and exchanging information among researchers.

Goals

Our goals are simple for S+ARRAYANALYZER:

1. Create rigorous statistical analysis that is easily initiated through either the menu or the command line.
2. Generate interactive tabular and graphical reporting that targets biologists and non-clinical statisticians.
3. Develop solution and extensible environments for customized applications.

For the latest information and support on S+ARRAYANALYZER, go to

http://www.insightful.com/support/ArrayAnalyzer

This contains information regarding Insightful efforts in the genomics and bioinformatics space.

Libraries

There are 15 function libraries and nine data libraries included in S+ARRAYANALYZER to assist in your analysis: affy, annotate, Biobase, edd, gcrma, genefilter, geneplotter, GOAnnoData, hgu133aAnnoData, hgu133acdf, hgu95av2AnnoData, hgu95av2cdf, KEGGAnnoData, LPEtest, marrayClasses, marrayInput, marrayNorm, marrayPlots, matchprobes, mgu74av2AnnoData, mgu74av2cdf, multtest, ROC, and vsn. All except LPEtest are ported from the Bioconductor project.

In this release of S+ARRAYANALYZER (version 2.0), new libraries have been added, including vsn, gcrma, and matchprobes. Note that these libraries are loaded automatically when you load S+ARRAYANALYZER.

The following shows a few examples of how S+ARRAYANALYZER can process your data:

- Analysis of Affymetrix data uses the Biobase and affy libraries for reading and normalizing data.
- Differential expression analysis uses the multtest and LPEtest libraries, and annotation is completed using the genefilter, geneplotter, and annotate libraries.
- Input and normalization of cDNA data uses the marrayClass, marrayInput, marrayNorm, and Biobase libraries.

Most of the ported libraries in S+ARRAYANALYZER are based on the Bioconductor 1.3 libraries, with some based on 1.4. Information on these libraries is located at

http://www.bioconductor.org

The other library, LPEtest, is provided by Insightful.

SUPPORTED PLATFORMS AND SYSTEM REQUIREMENTS

S+ARRAYANALYZER is supported on the following platforms:

- Windows 2000
- Windows 2003 Server
- Windows XP Professional

The minimum recommended system configuration is a Pentium III/1Ghz processor, at least 1GB of RAM, and an SVGA or better graphics card and monitor. You must have at least 850MB of free disk space for the typical installation (and, even if not installing on drive **C:**, an additional 2MB of free disk space on drive **C:** to unpack the distribution).

Installing and Running S+ARRAYANALYZER

To install S+ARRAYANALYZER, insert the S+ARRAYANALYZER CD, double-click the **setup.exe** file in the CD-ROM drive of your Windows Explorer, and follow the step-by-step installation instructions.

In S-PLUS, load the S+ARRAYANALYZER module from the command line by entering

```
> module(ArrayAnalyzer)
```

You can also load S+ARRAYANALYZER by choosing **File ▶ Load Module** and selecting **ArrayAnalyzer** from the menu.

To detach or unload S+ARRAYANALYZER, type

```
> detach("ArrayAnalyzer")
```

Online Help

S+ARRAYANALYZER also includes an online HTML Help system for all the available functions. After you have loaded the S+ARRAYANALYZER module, you can get help for any command by using the `?` or `help` function. For example, if want help on the `maBoxPlot` function, simply type

```
> module(ArrayAnalyzer)
> help(maBoxPlot)
```

at the Command line.

HTML Help

HTML Help in S-PLUS is based on Microsoft Internet Explorer and uses an HTML window to display the help files. You can access help on any function or GUI dialog in S+ARRAYANALYZER from the main menu by selecting **Help ▶ Available Help ▶ arrayanalyzer**. The HTML Help system includes a table of contents (organized by library), an index, and a **Search** button.

You can also get help on any S-PLUS function from either the command line or from **Help ▶ Available Help ▶ Language Reference**. If you need help on the S-PLUS GUI, click the **Help** button at the bottom of any dialog or navigate to **Help ▶ Available Help ▶ S-PLUS Help**.

Online Reference

In addition to the online help, you can access a pdf of the User's Guide by going to **Help ▶ Online Manuals ▶ ArrayAnalyzer User's Guide**. The *S+ARRAYANALYZER User's Guide* is particularly helpful for those new to S-PLUS and microarray analysis.

You can also access versions of the Bioconductor library pdfs in S+ARRAYANALYZER. The individual library pdfs are located at the top level of each library; for example, the Biobase library pdf is available at

splus62\library\Biobase\Biobase.pdf

Just double-click the file to launch the pdf.

Note these pdfs are current only up to this release. For updated information, please visit the Bioconductor Web site.

Technical Support

North, Central, and South America

Contact Technical Support at Insightful Corporation:

Telephone: 206.283.8802 or 1.800.569.0123, ext. 235, Monday-Friday, 6:00 a.m. PST (9:00 a.m. EST) to 5:00 p.m. PST (8:00 p.m. EST)

Fax: 206.283.8691

E-mail: support@insightful.com

Web: **http://www.insightful.com/support**

All Other Locations

Contact the UK office of Insightful Corporation:

5th Floor
Network House
Basing View
Basingstoke, Hampshire
RG21 4HG

Telephone: +44 (0) 1256 339800
Fax: +44 (0) 1256 339839
E-mail: shelp@insightful.com

If you purchased S+ARRAYANALYZER through our international distributor network, contact your local distributor:

http://www.insightful.com/contactus/internationaldistributor.asp

GENOMICS AND DIFFERENTIAL EXPRESSION

DNA microarrays are the most widely used tools in the analysis of gene expression and the study of functional genomics. Microarrays comprise gene-specific sequences (probes) immobilized to a solid state matrix, which are queried with mRNA from biological samples under study. Since many changes in cells are related to changes in mRNA levels for some genes, microarrays can be effectively used in a wide variety of applications including identification and validation of drug targets, characterization and screening of drug toxicities, exploration of biological pathways, and development of molecular diagnostics.

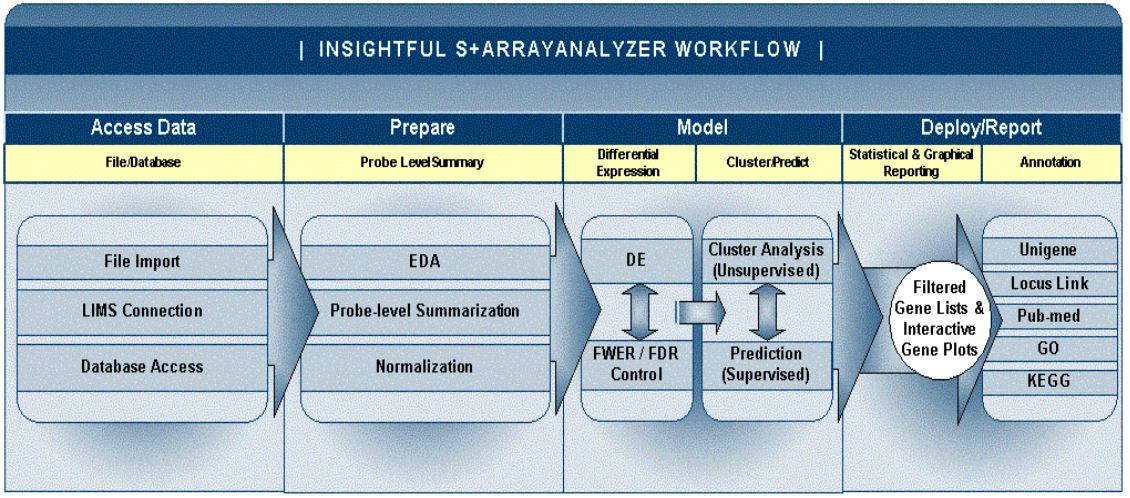


Figure 1.2: *Once data are obtained from a microarray experiment, several steps are required to prepare and analyze differential expression intensities and annotate the results with gene descriptions available in public databases like LocusLink or UniGene. This workflow shows the steps incorporated into the workflow of S+ARRAYANALYZER when doing differential expression analysis.*

Microarray technology is complex, and experiments using microarrays are resource-intensive. As such, there is an urgent need for rigorous statistical design and analysis of microarray experiments.

Statistical issues in microarray experiments include:

- Experimental design.
- Pre-processing (e.g., normalization).
- Differential expression testing.
- Clustering and prediction.
- Annotation.

All of these issues may be addressed with the use of modern statistical methods. Care is required, however, to perform the analysis correctly, and detailed collaborations between biologists and statisticians are a sound recipe for successful use of microarrays.

S+ARRAYANALYZER provides off-the-shelf functionality for microarray data analysis as well as a toolkit and development environment for custom microarray analysis solutions. Key packages from the Bioconductor project are included in S+ARRAYANALYZER:

http://www.bioconductor.org

Reports from microarray analysis such as summary gene lists and volcano plots are presented using S-PLUS Graphlets®, which facilitate interactive annotation of result summaries and allow you to share results via the Web.

MICROARRAY DATA

DNA microarrays are now widely used as a key experimental platform in drug discovery (e.g., functional genomics) and drug candidate evaluation (toxicogenomics). Their utility lies in the ability to simultaneously quantify the relative activity (or differential expression) of many genes under different biological conditions. Some common uses of microarray experiments are to

- Classify diseases and their subtypes.
- Identify and validate new targets for drug discovery.
- Improve understanding of biological processes.
- Evaluate drug candidates against drugs with known toxic side effects.
- Develop *personalized* treatment plans tailored to genotypes.

It is not our intention to discuss in depth the biology of microarrays. If you are new to this area, you should investigate the references listed at the end of the chapters in this manual, as most chapters provide references with detailed information. We give here a *brief* overview for those new to the area.

A microarray consists of a slide with genes (or active segments of genes) attached at *spots* on a regularly spaced grid. There may be anywhere from a few to tens of thousands of genes spotted on a single microarray which may occupy one or more slides. At each spot, one gene or an active segment of a gene is represented tens of thousands of times by cloning it and fixing all the duplicates to the spot on the slide.

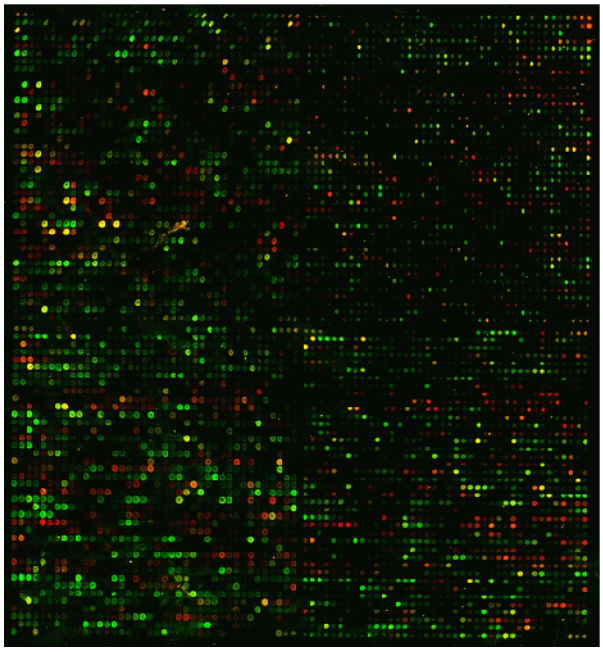


Figure 1.3: *Microarray experiments produce gene expression images like the one pictured here. These images must be converted to numbers (the quantification step) before analysis can proceed. Scanners like those from GenePix and Agilent produce raw intensity data files which form the starting point for differential expression analysis in S+ARRAYANALYZER.*

A gene expression experiment entails "washing" microarrays with concentrated cellular material and quantifying how much cellular substance *binds* to the gene spots. A great deal of binding at a spot indicates that gene is active in the cell, that is, the gene is being *expressed* in that cell or tissue. Knowing which genes are being expressed (or not expressed) and how that expression changes under different experimental conditions is of great importance in functional genomics and in developing new diagnostics, therapeutics, or treatment strategies.

S+ARRAYANALYZER is designed to work with data from different commercial microarrays. In particular it works with data from Affymetrix microarrays and from custom cDNA microarrays available through several suppliers. We describe in more detail the

differences between these two basic types of microarrays through examples in Chapters 2 through 4. Here we introduce them briefly to aid in understanding the examples that follow.

Affymetrix Arrays

Affymetrix® GeneChip®microarrays represent each gene with an oligonucleotide (25-mer) probe spotted at typically 11-20 pairs of spots (22-40 spots in all). Each probe pair consists of a spot for the probe, called a *perfect match* (PM) and a spot for a slight alteration of the probe, called a *mismatch* (MM). Non-specific binding may be accounted for by adjusting PM intensities to account for MM expression intensities.

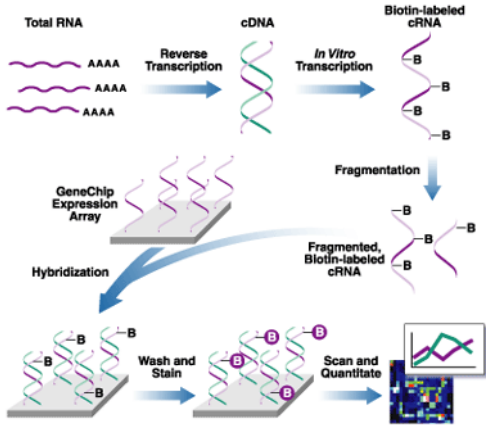


Figure 1.4: *Affymetrix's GeneChip® is a one-color oligonucleotide array. Mass produced, reliable, standardized microarrays like the GeneChip have help fuel the bioinformatics revolution.*

Affymetrix has revolutionized bioinformatics with its GeneChip technology.

To analyze Affymetrix expression data, all the expression values for each probe pair are first *summarized* by a *single* value. There are numerous ways to do this, and Affymetrix provides methods for such probe-level summarization in their MAS4 and MAS5 software. S+ARRAYANALYZER provides other methods for probe-level analysis which are discussed in depth in Chapter 3. Our example in Chapter 2 uses Affymetrix MAS 5 summary data.

Two-Color Arrays

cDNA or two-color microarrays are designed to compare *two* different samples (the experimental conditions) on *each* slide. Each sample is treated with a different color before it is added to the slide. Differential expression is computed as the difference between the color intensities of the two samples.

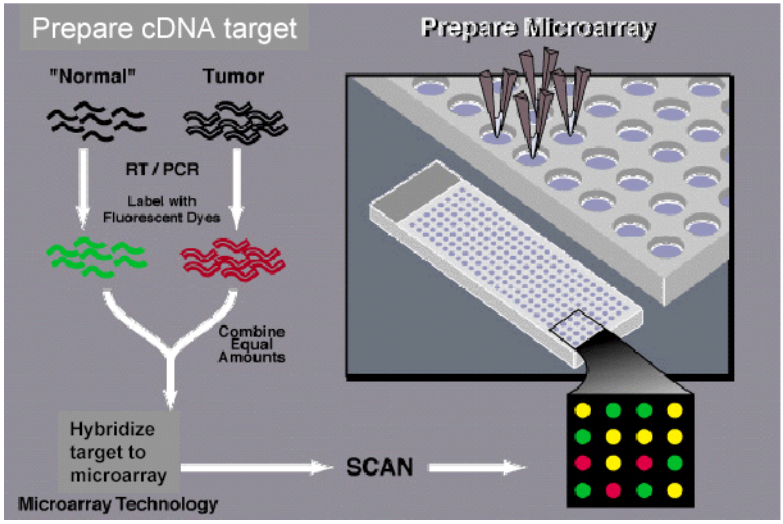


Figure 1.5: *Custom two-color cDNA microarrays compare treatments on each array by tagging them with different colors. This two-color design provides a way of estimating differential expression independent of chip to chip variability.*

cDNA microarrays may be customized, both gene content and layout, by the experimenter. Consequently the layout and gene content must be provided at the time of the analysis. This makes data import more complex than for Affymetrix chips for which there are many standard fixed layout descriptions. We provide a cDNA example in Chapter 4 to illustrate the steps involved in the analysis.

EXAMPLES: AFFYMETRIX MAS DATA

2

Affymetrix Data Analysis Workflow **16**

Experimental Design **17**
- Two-Sample Design 17
- One-Way Design 17

One-Way Design **18**
- Swimming Mice Data 18
- Importing Data 21
- Quality Diagnostics 31
- Normalization 36
- Differential Expression Analysis 40
- Gene Lists Management 48
- Annotation 50

From the Command Line **54**
- Importing Data 55
- Data Manipulation 57
- Normalization 59
- Differential Expression Testing 61

References **65**

AFFYMETRIX DATA ANALYSIS WORKFLOW

The entire process of analyzing gene expression data with Affymetrix MAS 4/5 or **.cel** file data can be done through the S+ARRAYANALYZER menus and dialogs. To obtain differential expression information from probe-level (**.cel** file) microarray data, we perform the following basic steps:

1. Import and filter the data.
2. Normalization including
   - Adjustment for background noise,
   - Mismatch correction,
   - Distribution-based normalization.
3. Summarize.
4. Differential expression analysis.
5. Annotation.

In addition, examining array quality and filtering out bad arrays and genes may be necessary and is typically done between import and normalization. Clustering is also a normal part of gene expression discovery and may be performed between all the major steps of the analysis.

Because MAS 4/5 data has already been corrected for background noise, mismatches and summarized, we can skip most of step 2 and all of step 3. However, if arrays have been analyzed with MAS 4/5 software, only simple normalization has been done, i.e., multiplying all expression values on a array by a single scalar such that the scaled mean expression values on each array are the same. This simple normalization is not enough to account for much extraneous variability (see Bolstad et. al., 2002) so we usually apply normalization procedures to MAS 4/5 data before analysis. The following examples demonstrate steps 1, 2, 4 and 5 for Affy MAS data.

EXPERIMENTAL DESIGN

Two-Sample Design

S+ARRAYANALYZER has an assortment of procedures for doing two-sample differential expression analysis. They include the t-test with or without the equal variance assumption, the Wilcoxon rank-sum test and several permutations tests. In addition the LPE test procedure, which produces improved error estimates when there is little replication in the design, is implemented for two-sample problems. All these procedures are suitable for doing simple comparisons between two groups - treatment versus control, tissue 1 versus tissue 2, etc. For more details about two-sample procedures see Chapter 7, Differential Expression Testing.

One-Way Design

A one-way design is an extension to the two-sample design. Instead of comparing two samples you compare multiple samples. For example treatment 1, treatment 2 and a control form a simple one-way design. The example we examine in this chapter is set up as a one-way design but primary comparisons can also be thought of a two-sample problems. We'll demonstrate both approaches to the problem.

ONE-WAY DESIGN

Swimming Mice Data

In this chapter, we step through the analysis of an experiment designed to improve understanding of the effect of chronic conditioning on the mass build-up of the left ventricular muscle of the heart. A study was conducted on eight-week-old mice which were regularly exercised by swimming. Over the course of 10 days, exercise was increased from 10 minutes twice a day to 90 minutes twice a day. Conditioning of the mice continued for 4 weeks. For controls, age-matched mice were used that did not exercise. For more details see

http://cardiogenomics.med.harvard.edu/groups/proj1/pages/swim_home.html

This experiment is a two-factor design with one factor indicating whether or not the mice were conditioned by swimming and the other factor indicating the amount of conditioning which ranged from 10 minutes to daily exercise for four weeks. Gene expression was measured in three-replicate sets (i.e., three mice) for each time point and each treatment (swim) - control (no swim) combination. The main hypothesis of interest involves discovering genes showing differential expression between the time points and the treatment conditions. It is believed that certain genes are involved in the enlargement of ventricular mass during chronic conditioning. The arrays and data files are listed in Table 2.1.

Table 2.1: *Experimental design and file association for the melanoma cancer study.*

Cond	Time	Rep	Array label	File Name
Swim	10min	1	Swim10min1	**Swim10min1**
Swim	10min	2	Swim10min2	**Swim10min2**
Swim	10min	3	Swim10min3	**Swim10min3**
Swim	2.5days	1	Swim2.5d1	**Swim2.5d1**
Swim	2.5days	2	Swim2.5d2	**Swim2.5d2**
Swim	2.5days	3	Swim2.5d3	**Swim2.5d3**

Table 2.1: *Experimental design and file association for the melanoma cancer study.*

Cond	Time	Rep	Array label	File Name
Swim	1wk	1	Swim1wk1	**Swim1w1**
Swim	1wk	2	Swim1wk2	**Swim1w2**
Swim	1wk	3	Swim1wk3	**Swim1w3**
Swim	2wks	1	Swim2wks1	**Swim2w1**
Swim	2wks	2	Swim2wks2	**Swim2w2**
Swim	2wks	3	Swim2wks3	**Swim2w3**
Swim	3wks	1	Swim3wks1	**Swim3w1**
Swim	3wks	2	Swim3wks2	**Swim3w2**
Swim	3wks	3	Swim3wks3	**Swim3w3**
Swim	4wks	1	Swim4wks1	**Swim4w1**
Swim	4wks	2	Swim4wks2	**Swim4w2**
Swim	4wks	3	Swim4wks3	**Swim4w3**
Swim	4wks+1wk	1	Swim4wks+1wk1	**Swim4w1w1**
Swim	4wks+1wk	2	Swim4wks+1wk2	**Swim4w1w2**
Swim	4wks+1wk	3	Swim4wks+1wk3	**Swim4w1w3**
NoSwim	10min	1	NoSwim10min1	**NoSwim10min1**
NoSwim	10min	2	NoSwim10min2	**NoSwim10min2**
NoSwim	10min	3	NoSwim10min3	**NoSwim10min3**
NoSwim	4wks	1	NoSwim4wks1	**NoSwim4w1**
NoSwim	4wks	2	NoSwim4wks2	**NoSwim4w2**

Table 2.1: *Experimental design and file association for the melanoma cancer study.*

Cond	Time	Rep	Array label	File Name
NoSwim	4wks	3	NoSwim4wks3	**NoSwim4w3**
NoSwim	4wks+1wk	1	NoSwim4wks+1 wk1	**NoSwim4w1w1**
NoSwim	4wks+1wk	2	NoSwim4wks+1 wk2	**NoSwim4w1w2**
NoSwim	4wks+1wk	3	NoSwim4wks+1 wk3	**NoSwim4w1w3**

These data have been obtained from the CardioGenomics PGA Public Data Web site located at

http://cardiogenomics.med.harvard.edu/public-data#

and are used here for the purpose of this example only. The data are available for free public download but have also been included with the distribution of S+ARRAYANALYZER.

Setting Up The Analysis

The fundamental question posed by this study is:

> *Which genes were involved (or expressed) during the buildup of muscle in the heart during conditioning over a period of four weeks?*

To answer this question, data for the conditioned mice were collected at 10 minutes, 2.5 days, 1 week, 2 weeks, 3 weeks, 4 weeks and 5 weeks. The fifth week was a week of rest for the mice so it is referred to as 4 weeks of conditioning plus 1 week of rest, or more simply, 4 weeks + 1 week. Control data was collected only at 10 minutes, 4 weeks and 4 weeks + 1 week. The suggestion is to match experimental mice (**Swim**) with control mice (**NoSwim**) as listed in Table 2.2.

Because the control data is duplicated (see Table 2.1), it is not appropriate to use a two-way ANOVA for the analysis even though that might be tempting. The reuse of data would produce inappropriately small error estimates for tests of conditioning in a

two-way analysis, increasing the Type I error. The proper setup for the analysis is a one-way ANOVA which tests for differences between the pairs of treatment conditions given by the rows of Table 2.2.

Table 2.2: *Matching experimental mice with control mice to conditioning by reusing control data.*

Test	Swim	No Swim
1	3 reps @ 10 min	3 reps @ 10 min
2	3 reps @ 2.5 days	3 reps @ 10 min
3	3 reps @ 1 wk	3 reps @ 10 min
4	3 reps @ 2 wks	3 reps @ 10 min
5	3 reps @ 3 wks	3 reps @ 4 wks
6	3 reps @ 4 wks	3 reps @ 4 wks
7	3 reps @ 4 wks + 1 wk	3 reps @ 4 wks + 1wk

For simplicity in this first example, we will focus on a subset of the experimental data. We will look only at the 3-week, 4-week and 4-week + 1 week data (i.e., tests 5, 6, and 7). The design is still unbalanced because of the reuse of the control data for the mice conditioned for three weeks. Consequently, the setup remains a one-way ANOVA with five factor levels, **Swim3wks**, **Swim4wks**, **Swim4wks+1wk**, **NoSwim4wks** and **NoSwim4wks+1wk**.

Importing Data

Start the example by reading in the arrays. To import Affymetrix data, from the main S-PLUS menu, select **ArrayAnalyzer ▶ Import Data ▶ From Affymetrix**.

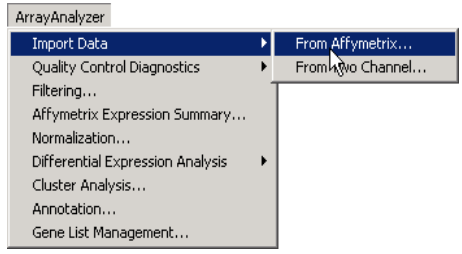


Figure 2.1: *Menu selection to import Affymetrix data.*

Import Affymetrix Data Dialog

Figure 2.2 shows the (**Import Data**) **from Affymetrix** dialog with the **File Selection** page displayed. The primary task of the import process associates data files with experimental conditions and selects variable columns that are used in subsequent analysis.

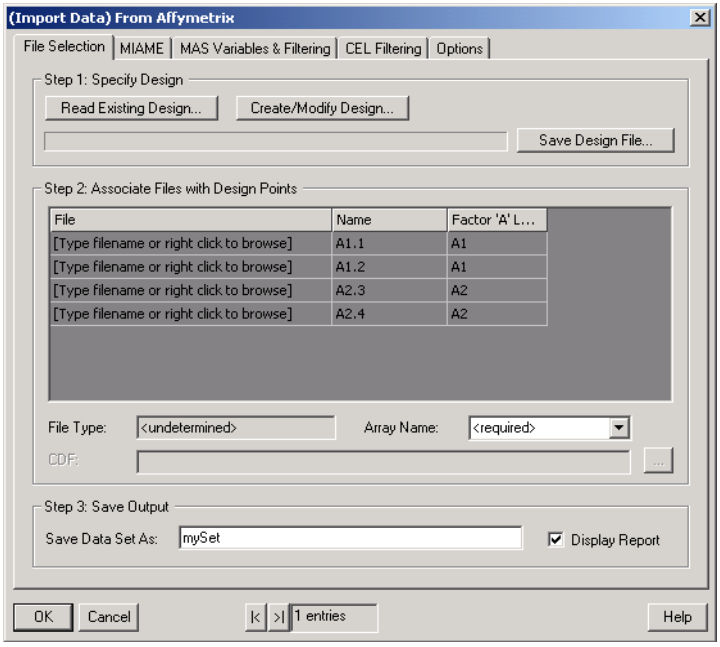


Figure 2.2: *The* ***(Import Data) from Affymetrix*** *dialog.*

The (**Import Data**) **from Affymetrix** dialog has five pages:

- **File Selection** This page *must* be completed in order to create a data object for continued analysis.
- **MIAME** Completing this page is optional but highly recommended because information on the **MIAME** tab is used for labeling tables and graphs.
- **MAS Variables & Filtering** This page has default settings depending on the type of data files (e.g., MAS4 or MAS5) you select. It also allows the selection of other variables which can be used for more general filtering by using the **Filtering** dialog.
- **CEL Filtering** This page allows a couple of options for filtering out spots not to be used in subsequent analyses when importing probe-level data.
- **Options** This page provides options for specifying the number of header lines to skip and the delimiter used in the data file.

Data import is accomplished in three steps:

1. Create the experimental design.
2. Associate files with design points.
3. Specify a name for saving the resulting data object.

Step 1: Create The Experimental Design

Before we can begin to associate data files with experimental conditions, we need to set up the experimental conditions in S+ARRAYANALYZER. The easiest way to do this is through the **Create/Modify Design** dialog. Open the **Create/Modify Design** dialog by clicking on the **Create/Modify Design** button on the **File Selection** page of the (**Import Data**) **From Affymetrix** dialog.

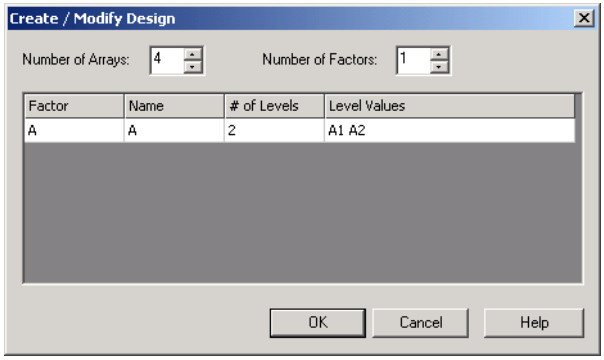


Figure 2.3: *The default* ***Create/Modify Design*** *dialog*

Use the **Create/Modify Design** dialog to specify the following:

1. The number of arrays to be read.
2. The number of factors in the experiment. Currently one or two are allowed.
3. The name, number of levels and level values for each factor.

Start by incrementing the **Number of Arrays** to 15. To modify the default factor **Name**, **# of Levels** and **Level Values**, type them into the appropriate field. For this example, click the row in the **Name** field once and enter **CondTime**, enter **5** for the **# of Levels** field, and click the **Level Values** field and enter **Swim3wks**, **Swim4wks**, **Swim4wks+1wk**, **NoSwim4wks** and **NoSwim4wks+1wk**.

For the one-way analysis of the Mouse Swimming data, we read 15 arrays and one factor with five experimental conditions. The resulting dialog is displayed in Figure 2.4.

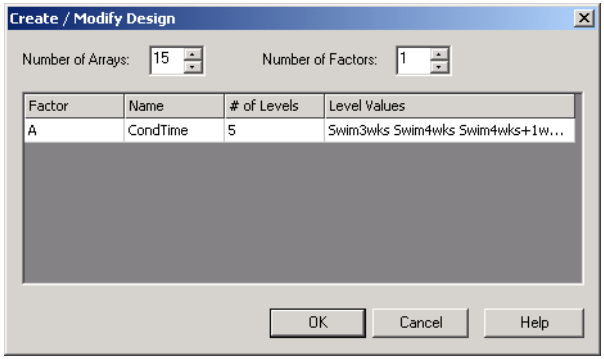


Figure 2.4: *The **Create/Modify Design** dialog with one-way experiment setup.*

Step 2: Associate Files With Design Points

Once the design is complete, click **OK** to copy it into the **File Selection** tab of the (**Import Data**) **From Affymetrix** dialog. Notice that the number of rows for the **File Selection** box is modified to match the number of arrays specified on the **Create/Modify Design** dialog. Furthermore, values for the factor levels have been written into the **Factor** columns to facilitate associating files with design points. If the experiment is balanced the factor level settings will be exactly as needed. However, the level values can be reset when the experiment is unbalanced or if you prefer an order different from the default.

The next step is to associate files with each design point. To do so right click in one of the file fields and browse to the location of your files.

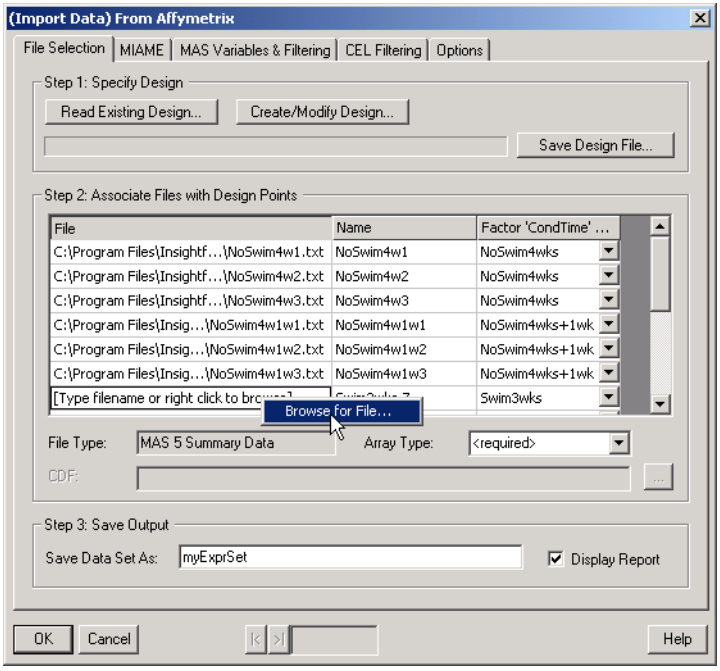


Figure 2.5: *Browsing for data files.*

You can find the swimming mice example data by navigating to your **splus62/module/ArrayAnalyzer/examples** directory and selecting the **Swim3w1.txt** file. Repeat for the other 14 **.txt** files, entering one file per field. A more efficient option is to select **Swim3w1.txt**, left-click, and scroll to the last sequential file in this list (**Swim4w3.txt**); repeat for the **NoSwim** data.

Alternatively, read the design file named **MouseSwimDesign.txt** in the examples directory to load the design and create file associations.

File Type

Note that the **File Type** entry, **MAS5 Summary Data** (listed below the files) is automatically detected once a file is selected. The dialog is designed to prohibit mixing file types.

Array Type

Array Type is a required field. You must select the name that corresponds to the Affymetrix array you used for your experiment. Some common examples are hgu133a and hgu95a. Click the drop-down button and select **mgu74av2**, as shown in Figure 2.6.

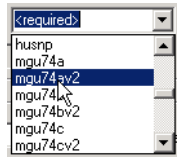


Figure 2.6: *Selecting **mgu74av2** as the Affymetrix array type.*

S+ARRAYANALYZER has pre-loaded the gene annotation information for arrays hgu95av2 and hgu133a. If you are using other arrays you may want to refer to the Chapter 6, Pre-Processing and Normalization to see how to load the annotation information for your array.

Step 3: Save The Data Object

To save the data object, type a name in the **Save Data Set As** field near the bottom of the dialog (**Step 3: Save Output**). Remember this name. It is used in other analysis steps such as quality checks, filtering and normalization. For our example, enter **MouseSwimExprSet** as the object name. The **Display Report** check box indicates whether or not to print summary information resulting from reading the data into an S-PLUS report window.

Figure 2.7: *Saving the imported data as **MouseSwimExprSet**.*

Saving the Design

Once you've entered all the information on this tab you can save it for later use by clicking the **Save Design** button at the top of the dialog. A **.txt** file is written to the directory of your choice with the number of factors, number of levels, repetitions and the full path file names and their associated factor levels.

Reading Designs

The *design* file can be reused for another experiment with the same or similar design by modifying the file locations and names and factor levels as needed. In fact, if you have many arrays in your experiment you can create a text file from scratch with all the design content and then read it with the **Read Existing Design** button. Reading the design in this way will fill the file name fields and their associated factor levels. For information about creating a design file see Appendix A, Creating a Design File.

MIAME Page

MIAME is an acronym for Minimal Information About a Microarray Experiment, and this information can be entered on the second page of the (**Import Data**) **From Affymetrix** dialog. This information is not required, but it is stored on the resulting object to identify the source of the data. Once you've entered MIAME information for any experiment, the first three fields are saved and are filled automatically the next time you open this dialog. The MIAME tab of the (**Import Data**) **From Affymetrix** dialog is shown in Figure 2.8.

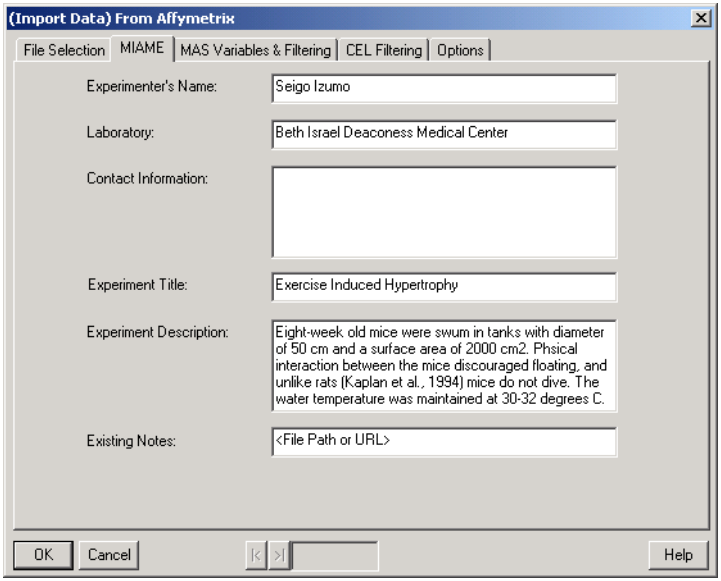

(Import Data) From Affymetrix
File Selection
MIAME
MAS Variables & Filtering
CEL Filtering
Options
Experimenter's Name:
Seigo Izumo
Laboratory:
Beth Israel Deaconess Medical Center
Contact Information:
Experiment Title:
Exercise Induced Hypertrophy
Experiment Description:
Eight-week old mice were swum in tanks with diameter of 50 cm and a surface area of 2000 cm2. Phsical interaction between the mice discouraged floating, and unlike rats (Kaplan et al., 1994) mice do not dive. The water temperature was maintained at 30-32 degrees C.
Existing Notes:
<File Path or URL>
OK
Cancel
Help

Figure 2.8: *Entering experiment information in the **MIAME** page.*

MAS Variables & Filtering Page

The third page of the (**Import Data**) **From Affymetrix** dialog is for variable and row selection. When reading MAS 4/5 data, this page is automatically filled. The **Probe Name** and **Expr. Intensities** drop-down fields are for selecting the columns in the data files corresponding to the probe names and expression intensities, respectively. Although it is possible to change the variables in the **Probe Name** and **Expr. Intensities** fields in this dialog, it is not recommended. These fields correspond to the columns read from the files and are used in subsequent analyses. The dialogs that follow in the data analysis, e.g., normalization and differential expression testing, expect expression data *without* control rows..

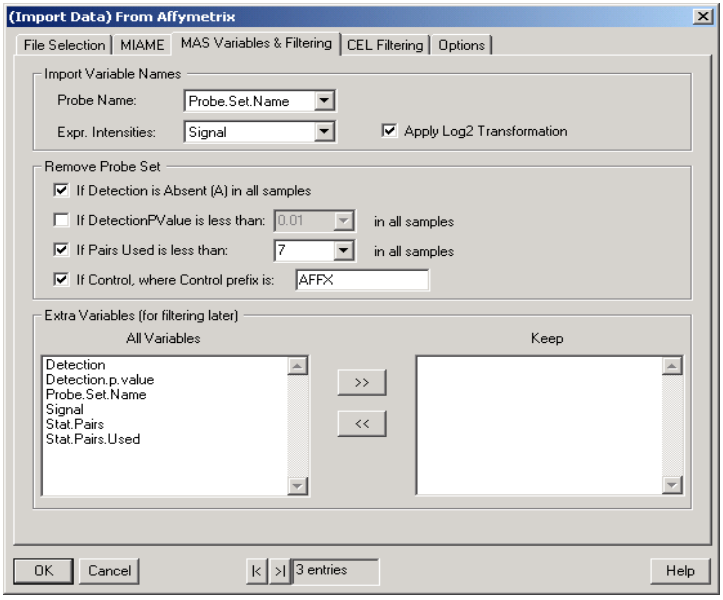


Figure 2.9: *The **Variable Selection & Filtering** page of the **(Import Data) From Affymetrix** dialog.*

Note the **Apply Log2 Transformation** check box which, by default, takes $\log_2$ of the expression intensities before saving them in the resulting object. The actual computation is $\log_2(E)$ if $E > 1$ and 0 if $E \leq 1$.

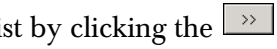
Fields you may be interested in changing are in the in the **Remove Probe Set** group. There are four options:

1. Remove probes that weren't detected in any sample.
2. Remove a probe if detection p-value is less then a specified value in all samples. Default p-value is 0.01.
3. Remove a probe if the number of pairs used in computing the summarized expression intensities is less than a specified value in all samples. The default value is seven. The maximum is the total number of probe pairs in a set, typically 11, 16 or 20.
4. Remove control probes that have a specified prefix. Default prefix is **AFFX**.

Keeping Extra Variables

The last group on the **MAS Variables & Filtering** tab allows you to select **Extra Variables** for filtering later. Use these extra variables for probe-set selection (or removal) before doing differential expression analysis.

Detection variable is required for a Genes Present Plot

If you want to do a **Genes Present Plot**, available on the **Quality Control Diagnostics** dialog, you *must* select the **Detection** variable from the **Extra Variables** list and move it to the **Keep** list by clicking the  button.

Other Options

The last page on the (**Import Data**) **From Affymetrix** dialog is the **Options** tab. The tab provides two options used during data import:

1. The number of header lines to skip in each file before reading the data. Normally this can be detected automatically but it is provided as an option for unusual cases where auto detection can not find the row with column names.
2. The delimiter separating the fields in each line of the data files. Normally these are tabs but alternative choices include comma, space, colon and semi-colon. Normally the delimiter can be detected automatically but this option is provided for unusual cases where auto detection can not determine the field delimiter.

Press **OK** when you have completed the dialog and the data are imported. It is now ready for use in S+ARRAYANALYZER.

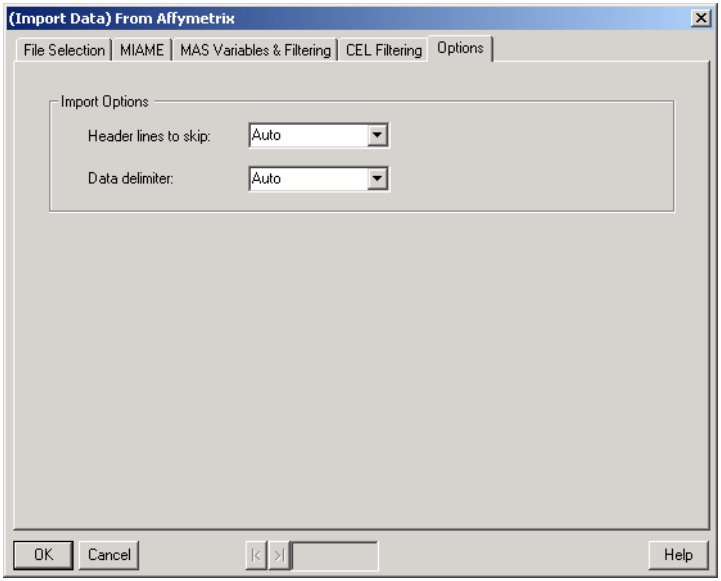


Figure 2.10: *The **Options** tab of the **(Import Data) From Affymetrix** dialog.*

Quality Diagnostics

Once the data is imported, open the **Quality Control Diagnostics** dialog to do quality checks. Open the dialog by clicking through the sequence **ArrayAnalyzer ▶ Quality Control Diagnostics ▶ Affymetrix** from the main S-PLUS menu bar.

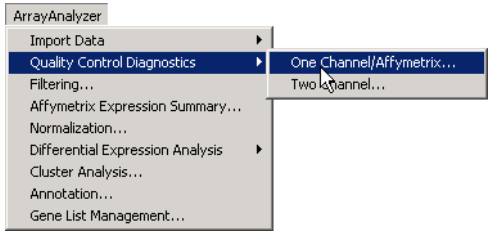


Figure 2.11: *Opening the **Quality Control Diagnostics** dialog.*

The resulting dialog provides a number of options for doing visual quality checks.

1. **Image Plot:** An image plot of each array. For Affy MAS data each vertical bar (or line when there are many genes) corresponds to a different gene.
2. **MvA Plot:** of replicates versus each other within treatment condition or versus the median expression computed for all replicates within each treatment condition. Options for the MvA plot are a) a scatter plot produced by drawing a random sample of genes and b) a hexbin plot of all genes where the hexagonal "points" are colored to give a sense of the density of points at each location.
3. **Genes Present Plot:** A simple barplot of percent of genes present for each array. Available only for MAS 4/5 data when the Detection column has been selected for filtering during the data import stage.
4. **Intensity Boxplot:** Boxplots of expression intensities for each array.
5. **RNA Degrad. Plot:** Plot of RNA degradation. Only available for probe-level (CEL) data.
6. **Principal Components Plot:** Plot of the first two principal components using treatment combinations (i.e., expression intensities for the entire array) as variables in the principal components analysis.

To generate the diagnostic plots, select **Affymetrix.Summary** in the **Show Data of Type** drop-down list and select **MouseSwimExprSet**. For the image plot set the **Page Layout** to 4 arrays per page. Select the **Genes Present Plot**, the **Intensity Boxplot** and the **Prin. Comp. Plot** check boxes.

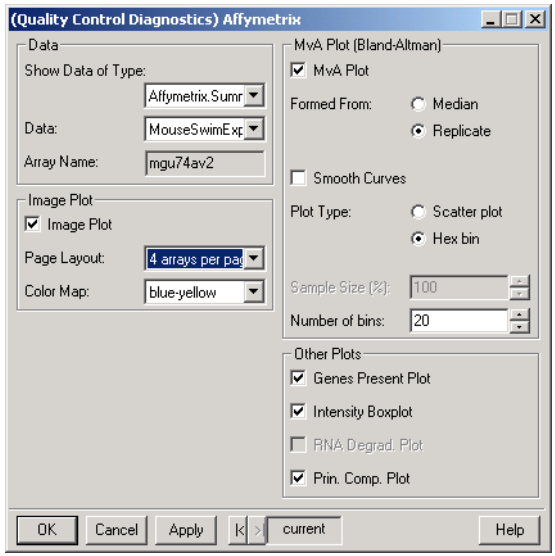


Figure 2.12: *Specifying options on the* ***Quality Control Diagnostics*** *dialog for the* ***MouseSwimExprSet*** *data set.*

The following set of graphs are examples of the QC Diagnostic plots for MouseSwimExprSet data set.

Figure 2.13 displays the image plots for four of the arrays in the study. Each gene corresponds to a different vertical line in the plot.

Figure 2.14 displays MvA plots for four replicate pairs. The plotting format uses hexbinning which shows the density of the data better than a simple scatter plot. Note the legend to the right describing how the colored hexagons relate to the density of points at each location.

Figure 2.15 displays the Genes Present Plot, a barplot representing the percent of genes detected on each array.

Figure 2.16 displays boxplots of expression intensities for all arrays in the study.

Figure 2.17 displays a plot of the first two principal components for all the arrays in the study. Each point in the plot corresponds to a different array. Different symbols correspond to different experimental conditions.

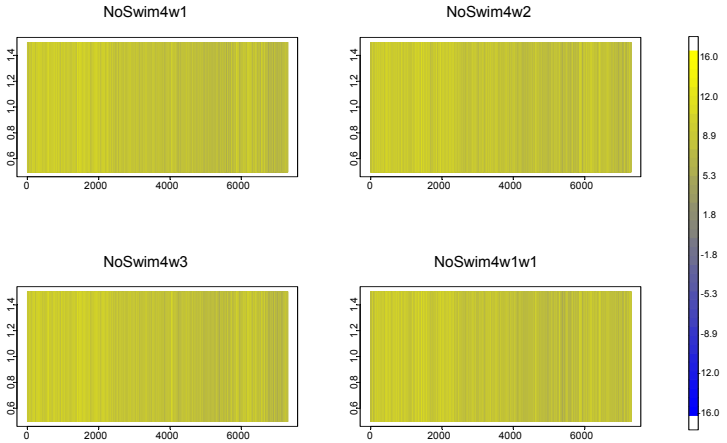


Figure 2.13: *Image Plot diagnostics for four of the arrays in the* ***MouseSwimExprSet*** *data set.*

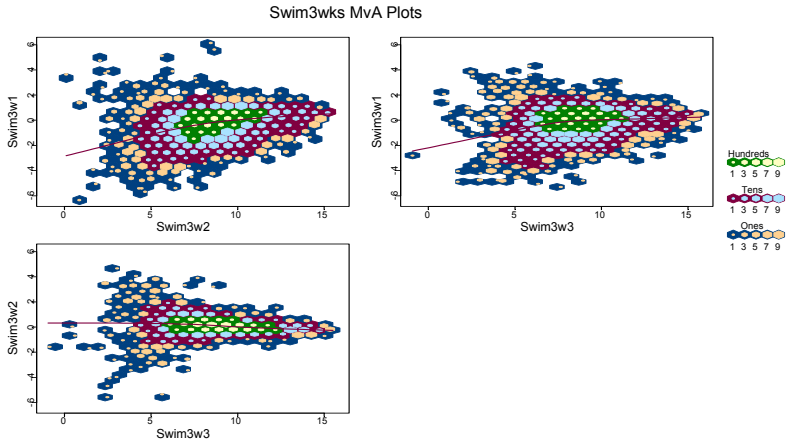


Figure 2.14: *A set of MvA hexbin plots for the three replicate pairs observed at week three of the Mouse Swimming study.*

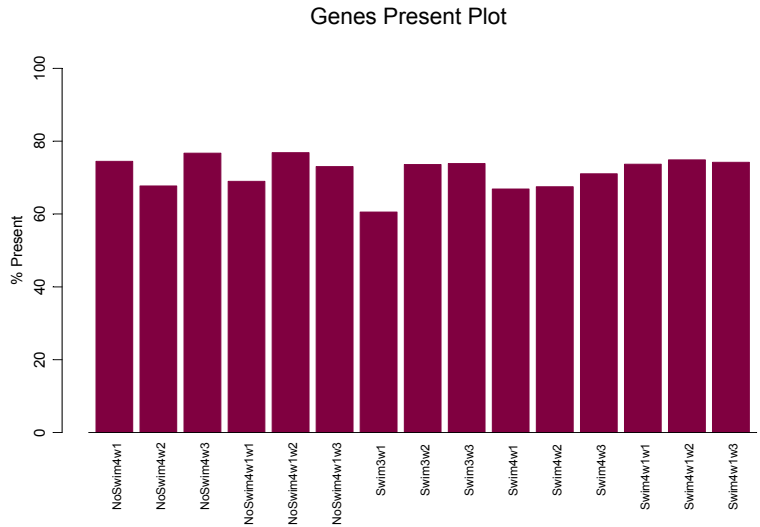


Figure 2.15: *Genes present plot of all arrays in the* ***MouseSwimExprSet*** *data set. Each bar corresponds to the percent of genes detected as indicated by the Detection variable in the Affymetrix MAS summary data.*

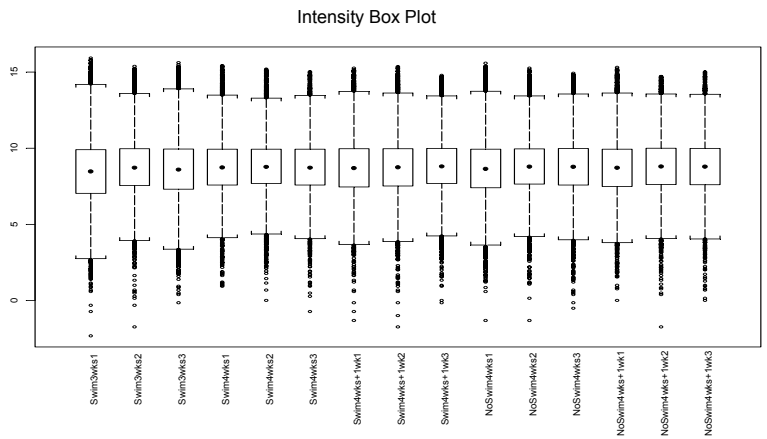


Figure 2.16: *Boxplots of expression intensities for all arrays in the* ***MouseSwimExprSet*** *data set. Each box represents the distribution of expression intensities for one array.*

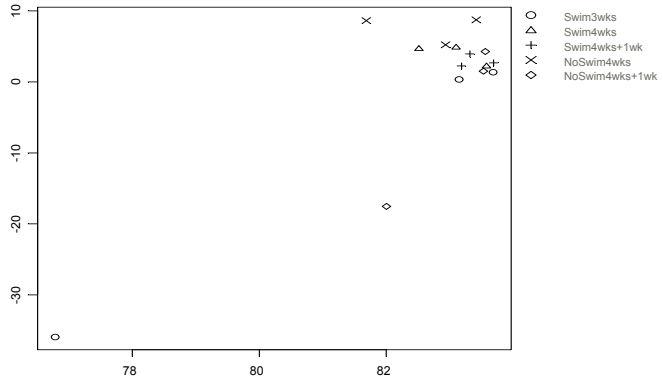


Figure 2.17: *Principal components plot of all arrays in the* ***MouseSwimExprSet*** *data set. Each point corresponds to a different array. Different symbols represent different experimental conditions.*

The diagnostic plots reveal nothing extraordinary about the arrays in the Mouse Swimming study so we'll move onto Normalization.

Normalization

Now that the data has been imported and checked for quality, we are ready to move to the next step of the analysis procedure: normalization. The **Normalization** dialog is designed to remove artifacts and systematic variation resulting from the measurement process. The goal is to remove variability not due to differential expression so that differential expression is estimated accurately for each gene. Note that we need to be careful not to normalize so aggressively as to wash out signal. Typically this is accomplished by normalizing within experimental conditions, although some forms of normalization may be comfortably applied across experimental conditions. For our swimming mouse example, this translates to normalizing within each conditioning level (Swim or NoSwim) and each observation time (3 wks, 4 wks and 4 wks + 1wk).

Normalization Dialog

To normalize the expression data, select **ArrayAnalyzer ▶ Normalization** from the main menu.

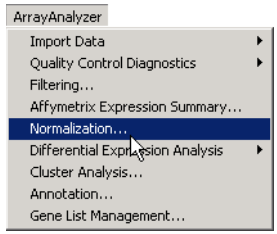


Figure 2.18: *Selecting the **Normalization** menu item.*

The Data Group

Show Data of Type

The **Normalization** dialog requires you to select the type of data you are working with. Click the drop-down button on the **Show Data of Type** field and select one of the choices. For the melanoma example, select **Affymetrix.Summary**.

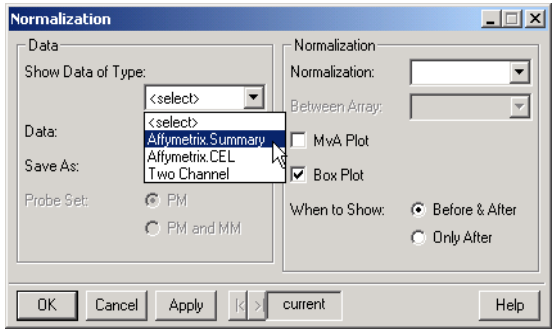


Figure 2.19: *Selecting the data type for normalization.*

Data

Click the drop-down button to right of the **Data** field and select the expression object created during the import step, **MouseSwimExprSet**.

Save As

Enter the object name for saving the normalized expression data in the **Save As** field. By default this is set to **MouseSwimExprSet.norm**, the name of the object you select in the **Data** field with**.norm** attached as a suffix.

The Normalization Group

In the **Normalization** group, set the **Normalization** field to **medianIQR**, select the **MvA** plot check box, select the **Box Plot** check box and click the radio button to select **Before & After** for pre-*and* post-normalization boxplots.

The normalization procedures for MAS 4/5 summary data are described in greater detail in Chapter 6, Pre-Processing and Normalization. For this example we select the default setting as **medianIQR**, which adjusts the location and scale of the data so expression values on all arrays have equal medians and equal inter-quartile ranges (i.e., the spread between the 25th and 75th percentiles is the same). The data normalization plots can be viewed before and after normalization or just after by selecting the appropriate choice.

Note that the **Probe Set** radio buttons are disabled for Affymetrix MAS data. These will be discussed in Chapter 3, Examples: Affymetrix Probe-Level Data.

Click **OK** or **Apply** to produce the normalized data and generate the pre- and post-normalization MvA pairs plots and box plots, as shown in Figures 2.20 and 2.21.

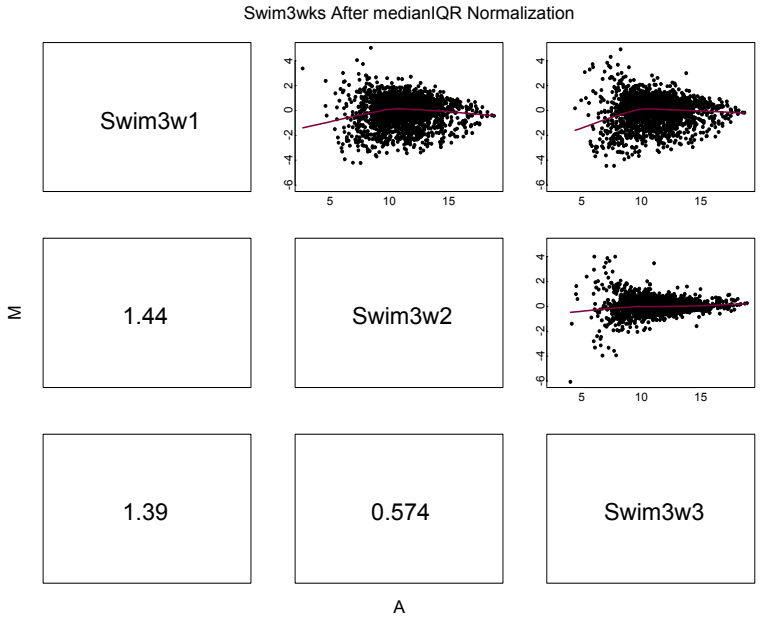


Figure 2.20: *MvA scatter plot matrix. Each plot is M vs. A for two arrays. To determine which arrays are used for each plot, go down vertically and left horizontally from the plot to the first array names you encounter. The numbers in the boxes below the diagonal are values of the interquartile range of M for the pair of arrays obtained by going up vertically and right horizontally to the first array names you encounter.*

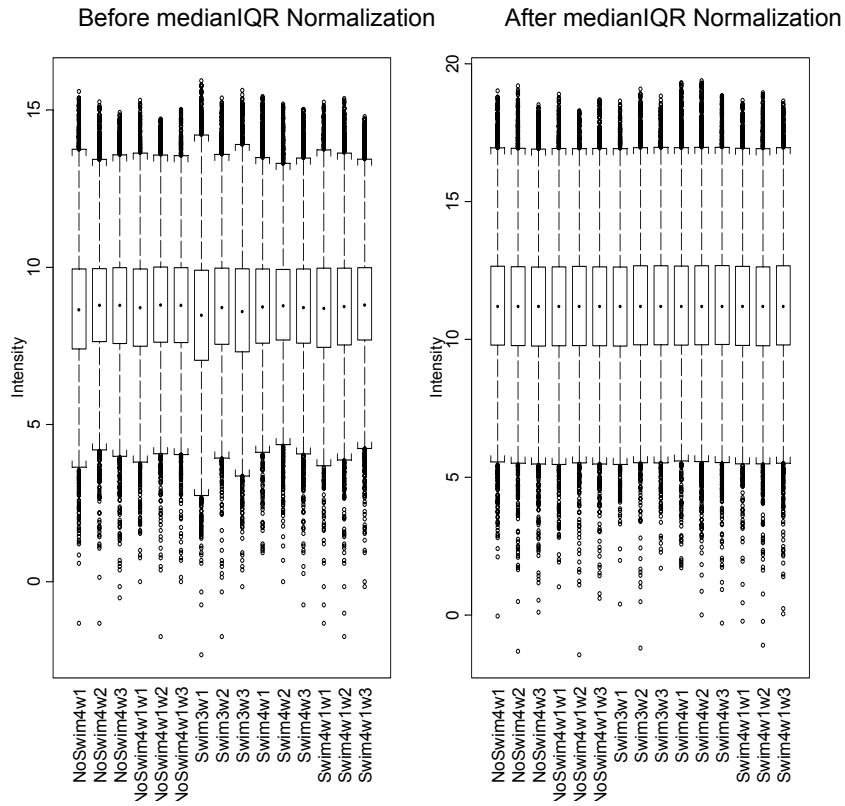


Figure 2.21: *Pre- and post-normalized boxplots of the swimming mice data.*

Additional MvA plots are generated for the other arrays both before and after normalization but they are not displayed here.

Differential Expression Analysis

For this example, we'll look at the analysis in two ways. The first applies a classical one-way analysis of variance (ANOVA) across the different levels of the factor. When setting up the ANOVA, we specify the type contrasts to test so we can produce the comparisons described in Table 2.2.

An alternate way to think of the analysis is as a set of two-sample problems as described in Table 2.2. This allows us to apply two-sample methods that may be more sensitive when there is little replication in the design.

The first analysis uses the ANOVA dialog.

ANOVA Dialog

We are now ready to do differential expression analysis for the one-way design. From the main menu, open the **ANOVA** dialog by clicking **ArrayAnalyzer ► Differential Expression Analysis ► ANOVA**. The procedures implemented in this dialog provide traditional ANOVA methods with a host of correction procedures to control the Type I Error rate. For more details on the **ANOVA** dialog and Type I Error correction procedures see Chapter 7, Differential Expression Testing.

To set up the **ANOVA** dialog, follow these steps:

1. In the **Show Data of Type** field, select **Affymetrix**.
2. In the **Data** field, select **MouseSwimExprSet.norm**.
3. The **Array Name** field should be automatically updated to **mgu74av2**.
4. In the **Contrasts** group confirm that the **Baseline** check box is checked.
5. From the **Baseline Level** drop-down list, select **NoSwim4wks**.
6. In the **Save As** field in the lower right-hand corner of the dialog type **MouseANOVANoSwim4wks**.
7. Click **OK** to compute the ANOVA results.

Once the **NoSwim4wks** baseline analysis completes, re-run it with baseline set to **NoSwim4wks1wk.** The additional analyses allow us to assemble all the significant genes for the contrasts listed in Table 2.2 for both of the baseline levels.

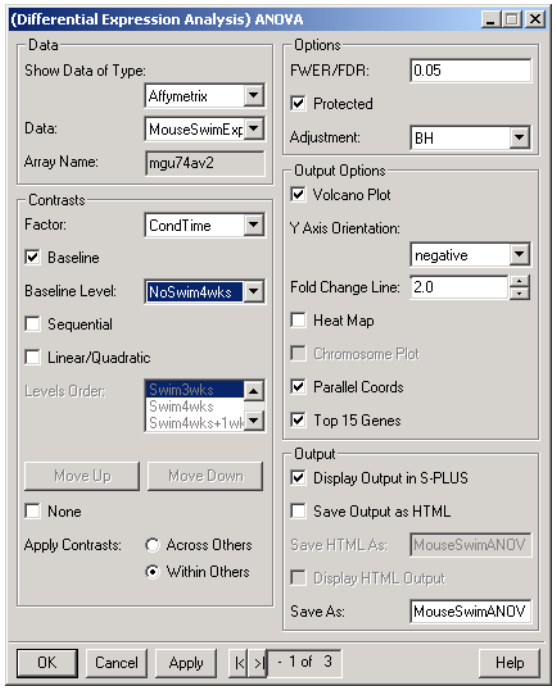


Figure 2.22: *The **ANOVA** dialog with settings for baseline contrasts.*

Options

The **Options** group allows you to set the family-wise error rate (FWER) or false discovery rate (FDR) to control the overall Type I error (false positive) rate based on adjusting individual test p-values to account for multiple tests. In our swimming mice example there are 7,624 genes so the Type I error is substantial without adjusting the p-values. See Chapter 7, Differential Expression Testing for more information on the error rate adjustment procedures.

There are many options for adjusting the p-values to achieve the FWER or FDR you want. Here we leave the default setting as **Bonferroni**.

There are four options in the **Output Options** group:

1. Volcano plot
2. Heat map
3. Parallel Coords
4. Top 15 Genes

Note that chromosome plots are not available for arrays other than **hgu95a**.

Figure 2.23 displays the volcano plot with Bonferroni FWER correction. Most of the p-values for the genes have been adjusted to one.

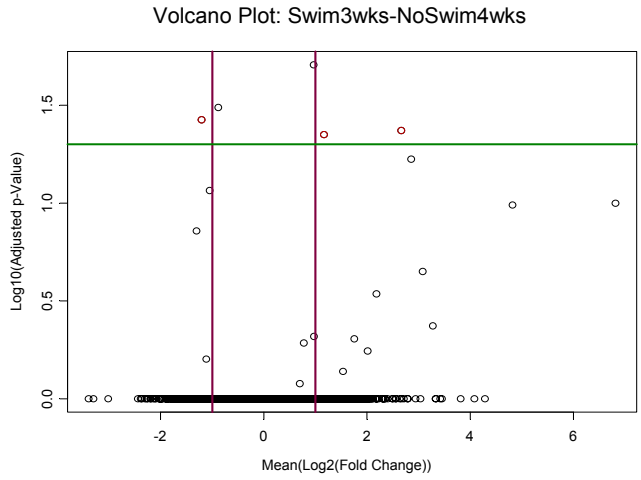


Figure 2.23: *Volcano plot resulting from Bonferroni FWER correction. The Bonferroni correction is very conservative and most of the p-value have been pushed to one.*

Let's try a less conservative adjustment procedure. We'll use the Benjamini and Hochberg FDR procedure which maintains a small percentage of false positives amongst only those genes which are significant. Select **BH** from the **Adjustment** drop-down list and save the resulting object as **MouseANOVANoSwim4wksBH**. Furthermore, turn on the HTML display by checking the **Save Output as HTML** and **Display HTML Output** check boxes. The resulting volcano plot is displayed in Figure 2.24. There are 21 significant genes in the plot resulting from the BH correction compared to five for the Bonferroni correction. Even with a four-fold increase in significant genes, the BH correction maintains a low false positive rate of 5% amongst the significant genes. This translates to, on average, only a single gene not really differentially expressed amongst those genes tagged as significant by the correction procedure.

Volcano Plot

A *volcano* plot displays the logarithm of p-value versus fold change, as shown in Figure 2.24. The vertical lines indicate fold change values of plus or minus one, and the horizontal line indicates a significant test p-value after doing the Benjamini-Hochberg correction. Points located in the upper, outer sextants are those with large absolute fold change and small (significant) p-value. Each of those points is active so you can click an individual point to access annotation information from Locus Link or GenBank.

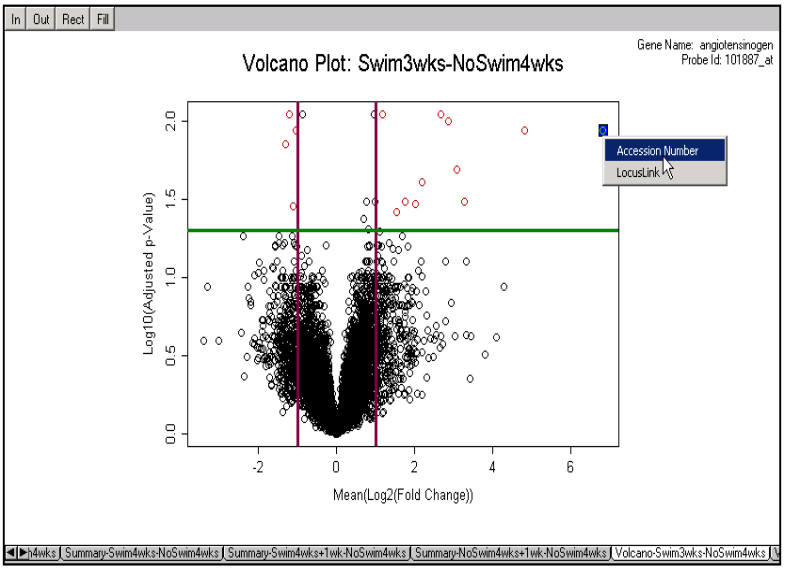


Figure 2.24: *A volcano plot, which is the logarithm of p-value versus fold change. The Benjamini-Hochberg FDR correction method was used for the swimming mouse data at an FDR of 0.05. The BH correction is less conservative than the Bonferroni procedure, yet maintains a small proportion of false positives amongst those genes tagged as significant.*

Heat Map

A *heat map* plot, shown in Figure 2.25, shows a 2-D plot of the most differentially expressed genes along the vertical axis versus the experimental conditions on the horizontal axis. This graph is also hyperlinked to the annotation information.

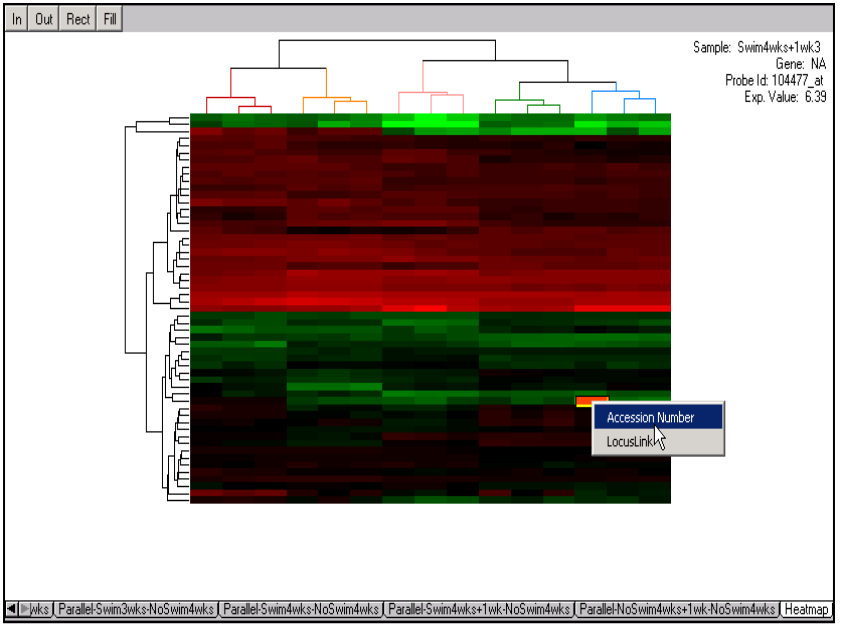


Figure 2.25: *A heat map plot shows differentially expressed genes as a function of experimental conditions.*

Graphical Annotation

Clicking one of the hyper linked points in either the volcano plot or the heat map pops up a menu for selecting the database to query for annotation information. Selecting either one opens an HTML page in your default web browser displaying a brief description of the gene with a hyper link to more detailed information. Figure 2.26 shows an example page from LocusLink with annotation for one of the differentially expressed genes in the swimming mice example.

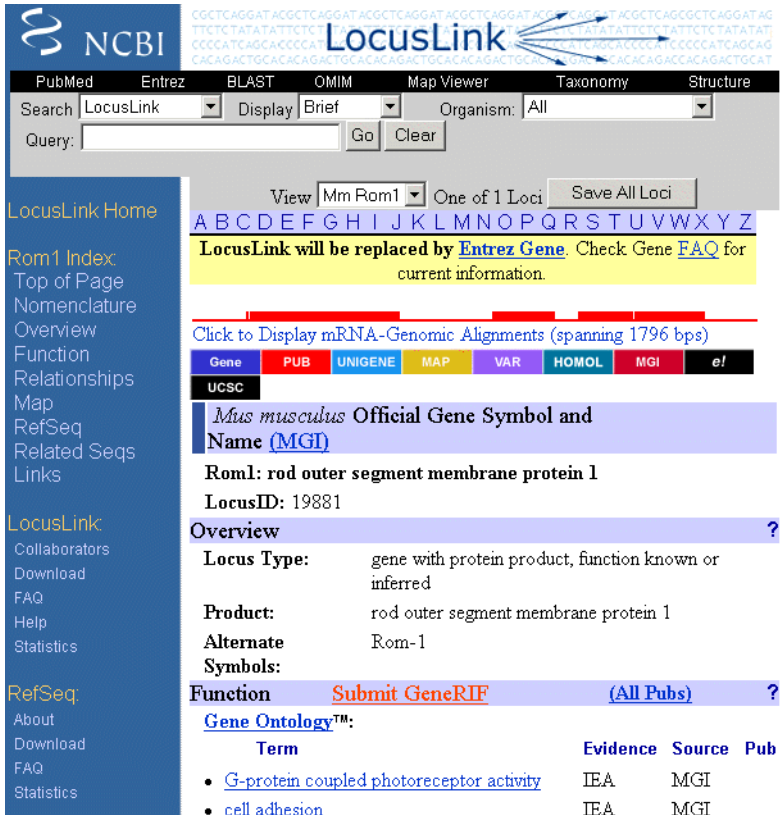


Figure 2.26: *Annotation information from LocusLink.*

The LPE Test

Alternative to the **ANOVA** dialog which tests multiple contrasts at once, you can test a particular contrast of interest directly with one of the two-sample test procedures. These are made available in two dialogs - the **Two-Sample Tests** and the **LPE Test** dialogs. To demonstrate the idea, open the **LPE Test** dialog by clicking **ArrayAnalyzer ► Differential Expressing Analysis ► LPE Test**. In the resulting dialog select **Affymetrix** data type, the **MouseSwimExprSet.norm** data set, and the levels for comparison in **Compare Level 1** and **Compare Level 2** fields. Save the result in **MouseSwimLPETest**. The settings are displayed in Figure 2.27.

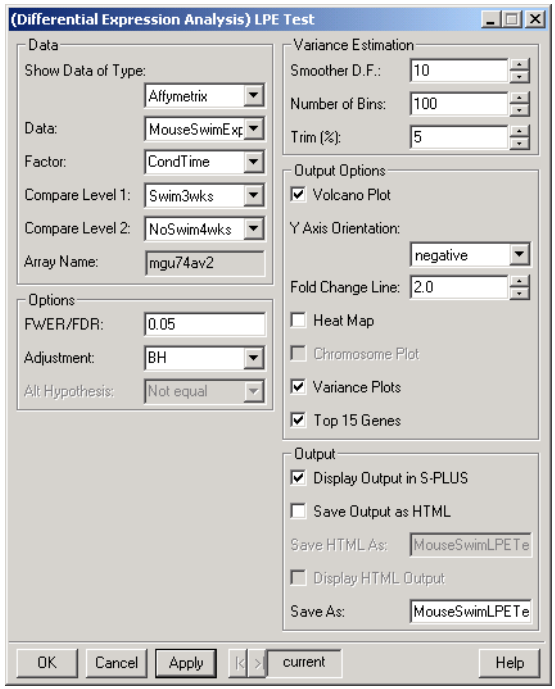


Figure 2.27: *The **LPE Test** dialog.*

Setting the adjustment to BH as we did for the ANOVA yields 22 significant genes when comparing:

```
Test Input: MouseSwimExprSet.norm
Test Input Class: exprSet
ChipName:  mgu74av2
Test:  LPE t-test
Adjustment Procedure:  BH
FWER:  0.05
```

```
Number of Tests: 7284
Number of Significant Expressions - overall: 40
    Number of Significant Expressions for contrast Swim3wks-
NoSwim4wks : 40
```

Gene Lists Management

The **Gene List Management** dialog allows you to merge gene lists from testing different contrasts in an analysis. We set up the Mouse Swimming experiment as a one-way design and we want to test a subset of the contrasts listed in Table 2.2. We do so by running two different ANOVA's choosing a different baseline each time: **NoSwim4wks** and **NoSwim4wks+1wk**. The same strategy could be applied to the entire data set including **NoSwim10min** as a baseline as well.

When the ANOVA runs it compares all the levels to the baseline. Consequently, more contrasts are computed than needed. The **Gene List Management** dialog allows you to select the contrasts you are interested in and merge the gene lists together.

Open the **Gene List Management** dialog by clicking **ArrayAnalyzer ▶ Gene List Management.** Now in **Data Group 1** select **DiffExpr Test** from the **Show Data of Type** drop-down list. Then select the **MouseSwimANOVABs4wkBH** ANOVA object from the **Data** drop-down list and select one of the contrasts listed in Figure 2.2. We start with **Swim3wks - NoSwim4wks**.

Repeat this selection for the other data group. Pick the same data type in **Data Group 2** and the same data object but choose **Swim4wks - NoSwim4wks** for the contrast. Now check the **Union** radio button in the **Output Options** group, confirm that the **Venn Diagram** check box is checked and type in SwimGeneList4wks as the name of the object to save the resulting gene list.

Figure 2.28 displays the finished dialog and Figure 2.29 displays the resulting Venn Diagram showing the resulting union of three gene lists.

Note that by selecting the Intersection radio button we can produce the intersection of gene lists with different baseline times. This allows us to discover early-indicator genes that express early in the heart muscle build-up process and continue expressing throughout.

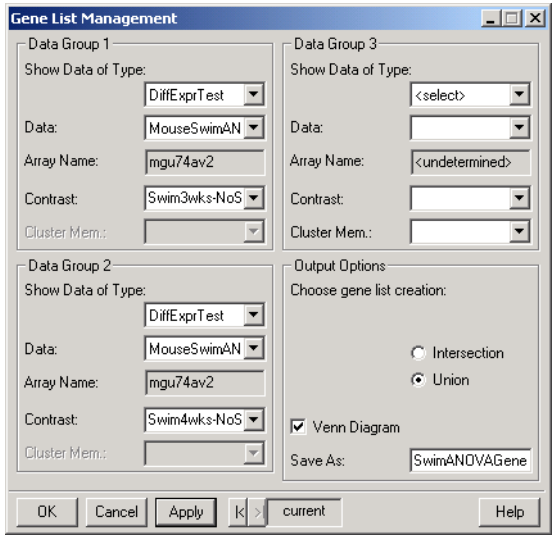


Figure 2.28: *Setting up the Gene List Management dialog for merging gene lists.*

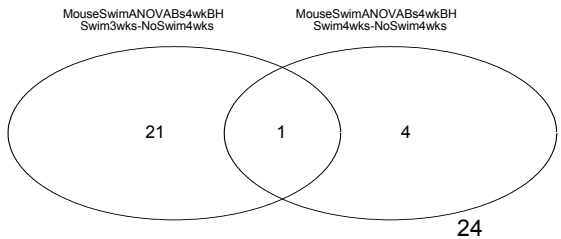


Figure 2.29: *Venn Diagram resulting from merging two gene lists.*

Also, you can compare the results between the ANOVA and LPE Test results for the **Swim3wks - NoSwim4wks** contrast by comparing their gene lists. Select the two differential test objects in the **Gene List Management** dialog and select the above contrast. The results are displayed in Figure 2.30.

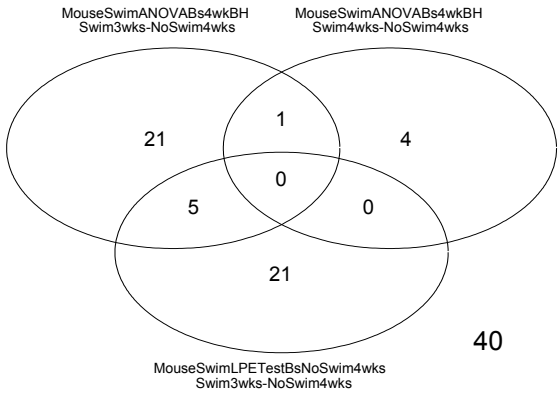


Figure 2.30: *Comparison of gene lists generated by ANOVA and LPE Test procedures.*

Annotation

In addition to the annotation described in section Graphical Annotation on page 46, there is a dialog with more general annotation capabilities. To open the **Annotation** dialog, click **ArrayAnalyzer ► Annotation**. The resulting dialog has two tabs:

1. **General Options**.
2. **Filtering Options**.

There are a number of options available on the **Annotation** dialog, but, for this first example, we'll focus on generating the annotation information for the significant genes in **MouseSwimANOVABs4wkBH**. For a more thorough description of the **Annotation** dialog see Chapter 9, Annotation and Gene List Management.

To annotate the significant genes identified in **MouseSwimANOVABs4wkBH**, select **DiffExprTest** from the **Show Data of Type** list and select **MouseSwimANOVABs4wkBH** from the **Data** list. Note the public annotation databases available from the

GUI in the **General Annotation** group. For the example we leave the three defaulted databases checked. Figure 2.31 displays the resulting settings.

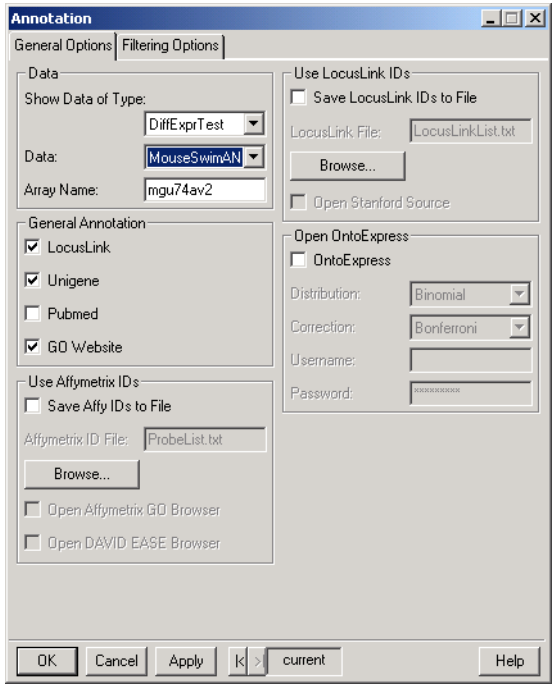


Figure 2.31: ***General Options*** *settings for annotation of the genes identified by the* ***MouseSwimANOVABs4wksBH*** *ANOVA.*

Now on the **Filtering Options** tab in the **Contrast Filtering** group select the **Swim3wks-NoSwim4wks** contrast and check the **Significant genes** checkbox. Clicking the **Recalculate** button in the **Gene Sort Order Options** group will show you how many genes are selected by the filtering. Note the **Limit number of genes to** field which puts a cap on the number of gene ID's that will be sent to the databases for annotation extraction. Figure 2.32 displays the resulting settings.

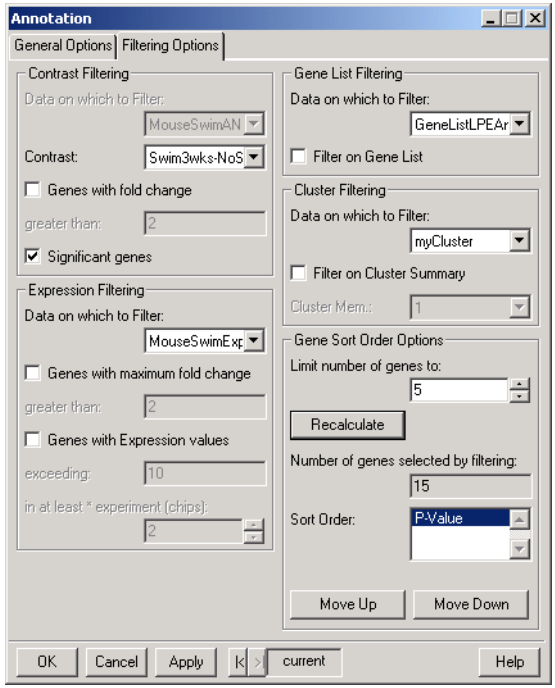


Figure 2.32: *Filtering Options settings for annotation of the genes identified by the* ***MouseSwimANOVABs4wksBH*** *ANOVA.*

Clicking OK now will generate the annotation lists from the databases selected. Figure 2.33 displays a summary table for the top significant genes identified by the ANOVA tests for the **Swim3wks-NoSwim4wks** contrast.

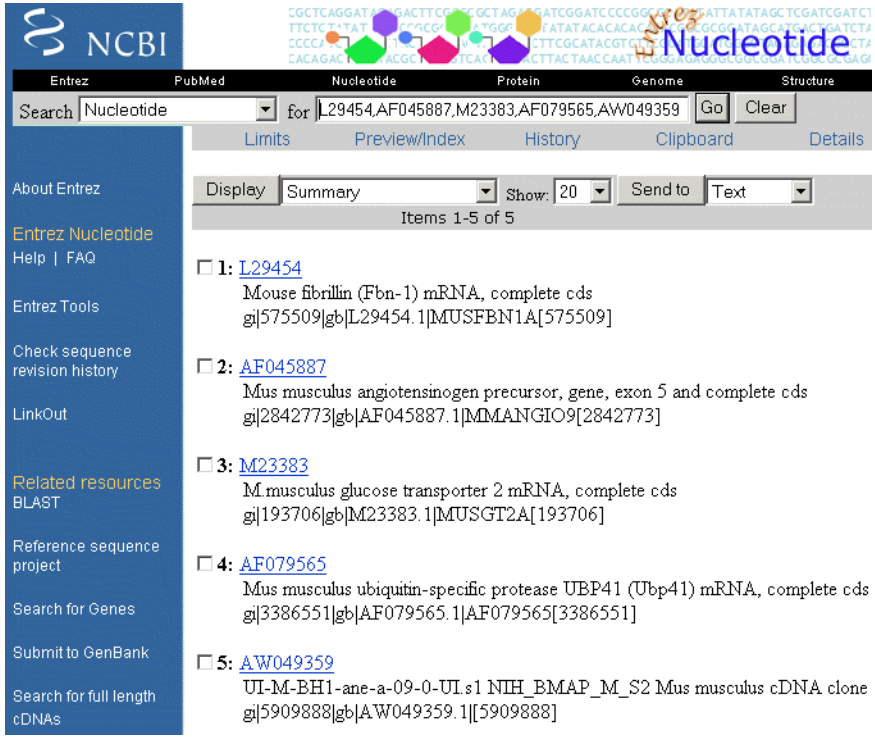


Figure 2.33: *Annotation summary from Unigene.*

FROM THE COMMAND LINE

All of the analysis done through the GUI can be done from the S-PLUS command line. Having access to the command line adds great flexibility to the set of features available through the S+ARRAYANALYZER GUI and opens the door to additional analyses. The flexibility and feature-rich S-PLUS language make it an ideal platform for exploratory analysis, statistical testing and modeling of gene expression data.

This section is designed to expose you to the critical functions for differential expression testing of microarray data. If you have no interest in running your analyses from the command line, you can skip this section.

In this section, we step through the analysis of an experiment using an MM5 melanoma cell-line in which a gel matrix that simulates the *in-vivo* cellular condition (and progression) of melanoma was added for 0 and 24 hours later (Fox et al., 2001). This simple experimental design thus involved one-factor (matrix condition) at two levels (0 and 24 hours) replicated twice at each time point. The main hypothesis of interest involves discovering genes showing differential expression at the two time points because these genes are believed to be relevant to tumor invasion and metastasis. The chips and data files are in Table 2.3.

Table 2.3: *Experimental design and file association for the melanoma cancer study.*

Experimental Condition	Repetition	chip label	File Name
0 hours	1	**cga**	**0hA.csv**
0 hours	2	**cgb**	**0hB.csv**
24 hours	1	**cg24a**	**24hA.csv**
24 hours	2	**cg24b**	**24hB.csv**

Importing Data

S-PLUS has several command line functions for importing data as well as a very general facility for importing data through the GUI. It is worth spending a little time importing data through the S-PLUS GUI because the facility is quite general and easy to use.

General GUI Import

To import a data file though the GUI, go to **File ▶ Import Data ▶ From File**. When the dialog opens, select the **File Format** and then browse for files. Figure 2.34 shows the **Data Specs** page of the S-PLUS **Import From File** dialog. You specify the name of the saved data object in the **Data Set** field.

If there is any header information in the file you need to specify **Start row** so the header information is skipped. You do that on the **Options** page of the **Import From File** dialog. Figure 2.35 shows the location of the **Start row** field (See the position of the mouse cursor.)

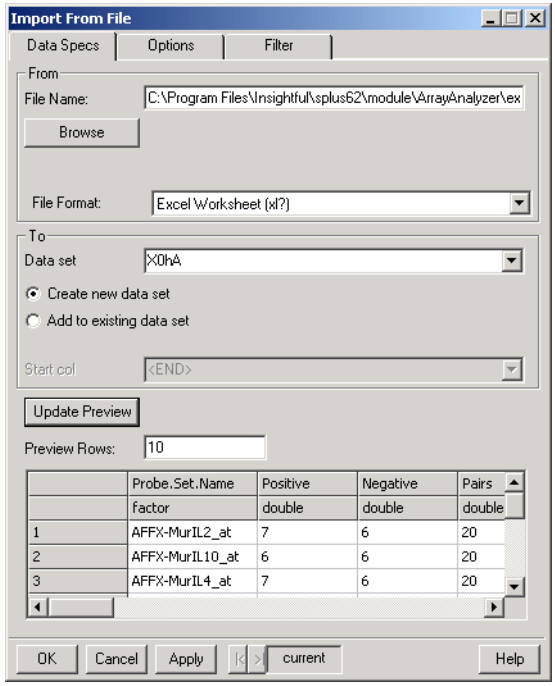


Figure 2.34: *Importing data through the S-PLUS GUI.*

Command Line Import

The command line equivalent to the **Import From File** dialog is the `importData` function. The critical arguments are:

file a character string specifying the name of the file to import.

type a character string specifying the type of file to import. Possible values are listed here; the case of the character string is ignored.

startRow an integer specifying the first row to be imported from the data file. This argument is available only when importing data from a spreadsheet.

To see descriptions of other arguments check the help file.

```
> help(importData)
```

For the melanoma data, we do a series of four `importData` commands to read the four files:

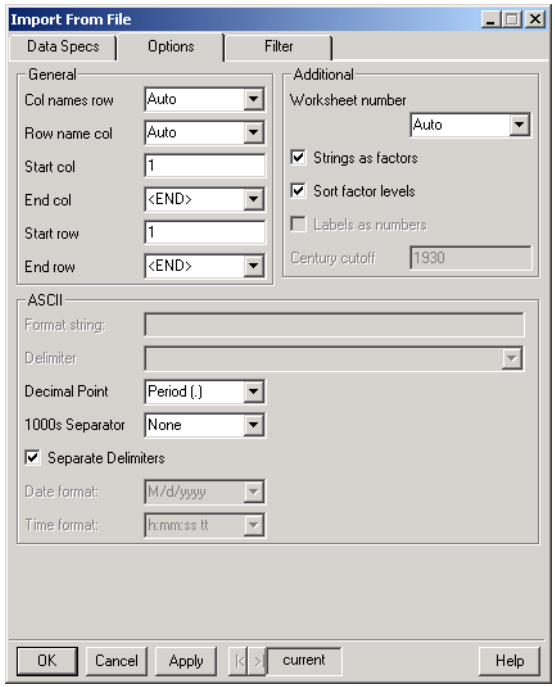


Figure 2.35: *Setting the **Start row** option of the **Import From File** dialog.*

```
> cga <- importData(paste(getenv("SHOME"),
     "/module/ArrayAnalyzer/examples/0hA.csv",sep="/"),
      type = "ASCII")
> cgb <- importData(paste(getenv("SHOME"),
     "/module/ArrayAnalyzer/examples/0hB.csv",sep="/"),
      type = "ASCII")
> cg24a <- importData(paste(getenv("SHOME"),
     "/module/ArrayAnalyzer/examples/24hA.csv",sep="/"),
```

```
       type = "ASCII")
> cg24b <- importData(paste(getenv("SHOME"),
     "/module/ArrayAnalyzer/examples/24hB.csv",sep="/"),
      type = "ASCII")
```

The resulting objects are data frames so you can do whatever you want to do in the way of data summaries and exploratory plots. First we'll take care of some organizational details such as converting all column names to lower case to make typing easier:

Data Manipulation

```
### Change column names to all lower case
> names(cga) <- casefold(names(cga))
> names(cgb) <- casefold(names(cgb))
> names(cg24a) <- casefold(names(cg24a))
> names(cg24b) <- casefold(names(cg24b))
```

Now lets find the control spots. All the arrays are the same so we can work off one of the data sets.

Extracting Probe Names and Finding Controls

```
###  Extract probe names
> cg.probes <- cga$probe.set.name

### Find control spots
> prefix <- substring(cg.probes, 1, 4)
> controls <- prefix == "AFFX"
```

You can eliminate genes with few spots used in their summarization by a simple subset operation. We repeat it for each array object.

Removing Genes With Few Good Spots

```
### Set avg.diff to missing wherever pairs.used < 7

> cga$avg.diff <- ifelse(cga$pairs.used<7,NA,cga$avg.diff)
> cgb$avg.diff <- ifelse(cgb$pairs.used<7,NA,cgb$avg.diff)
 < same for the other two arrays >
```

One example exploratory plot is the comparison of control and non-control spots. We can generate boxplots as follows:

Comparing Controls and Non-controls

```
> par(mfrow = c(2,2))
> boxplot(list(controls = logb(cga$avg.diff[controls], 2),
      noncontrols = logb(cga$avg.diff[!controls], 2)),
      ylab = "Log 2 Expression Intensities")
> title("0 hr, Replicate A")
```

The above expression creates a pair of boxplots for the first 0-hour replicate. By repeating the commands for the other three arrays we produce the remaining plots in Figure 2.36.

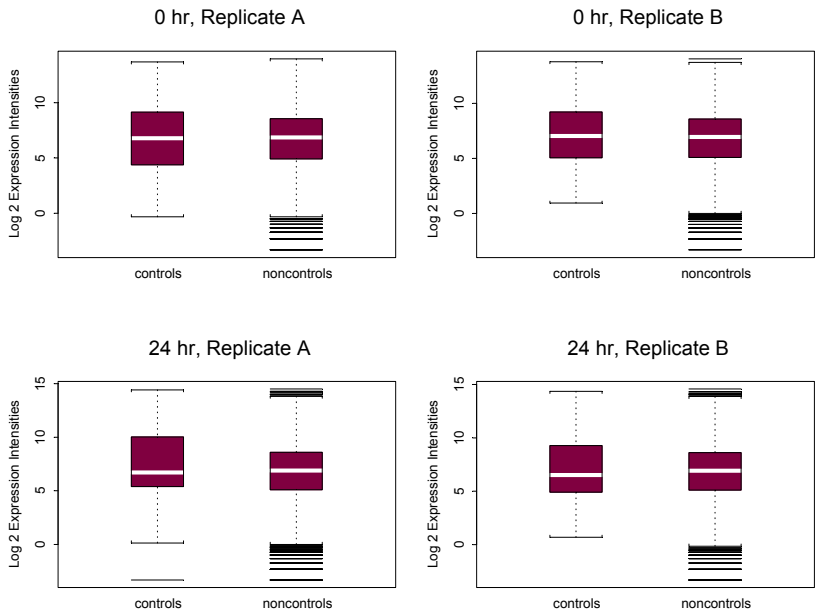


Figure 2.36: *Boxplots of control versus noncontrol spots for the melanoma data.*

Creating an Expression Intensity Data Frame

Now extract the expression intensities from each array in preparation to normalization and differential expression testing. Extract the **avg.diff** column and add it to a data frame named **CG**. For MAS5 data, this is the signal column.

```
> CG <- data.frame(CGa = cga$avg.diff, CGb = cgb$avg.diff,
          CG24a = cg24a$avg.diff, CG24b = cg24b$avg.diff)
```

Logging Expression Intensities

Compute the base 2 log transformation of the intensity values as follows. Any intensity values less than one will be negative or missing after taking logs so we set them explicitly to one in the `ifelse` function call.

```
### Threshold and log adjusted average differences
> LCG <- CG
> for (i in names(LCG)) LCG[[i]] <-
        logb(ifelse(CG[[i]] <= 1, 1, CG[[i]]), base = 2)
```

Removing Controls

Now remove the control rows from the expression data frame.

```
### Remove controls
> LCG <- LCG[!controls,]
```

Normalization

We normalize the intensity values by adjusting within array expression values to a common median and inter-quartile range using the **medianIQR.norm** function

```
### Normalize using medianIQR
> LCG.N <- data.frame(medianIQR.norm(LCG))
```

Compute a summary of the resulting logged and normalized expression intensities as follows:

```
### Summarize Non-controls
> summary(LCG.N)
       CGa                CGb               CG24a
 Min.: 0.5415185   Min.: 0.000000   Min.: 0.5071245
 1st Qu.: 3.2774186   1st Qu.: 3.277464   1st Qu.: 3.2121624
 Median: 6.6429682   Median: 6.642968   Median: 6.6429682
 Mean: 6.0522140   Mean: 5.964245   Mean: 6.0087211
 3rd Qu.: 8.8044455   3rd Qu.: 8.804491   3rd Qu.: 8.7391893
 Max.:14.6903453   Max.:15.151559   Max.:15.1604125
 NA's: 8.0000000   NA's: 8.000000   NA's: 8.0000000

      CG24b
 Min.: 0.5725788
 1st Qu.: 3.1813881
 Median: 6.6429682
 Mean: 5.9970227
 3rd Qu.: 8.7084150
 Max.:15.1399289
 NA's: 8.0000000
```

Note the missing values in the summary output. We'll have to take care of those before doing differential expression testing, but first let's plot normalized and unnormalized data for comparison.

```
### Before and after normalization boxplots
> par(mfrow = c(1,2))
> boxplot(LCG, style.bxp = "att")
> title("Before Normalization")
> boxplot(LCG.N, style.bxp = "att")
> title("After Normalization")
```

The resulting boxplots are displayed in Figure 2.37.

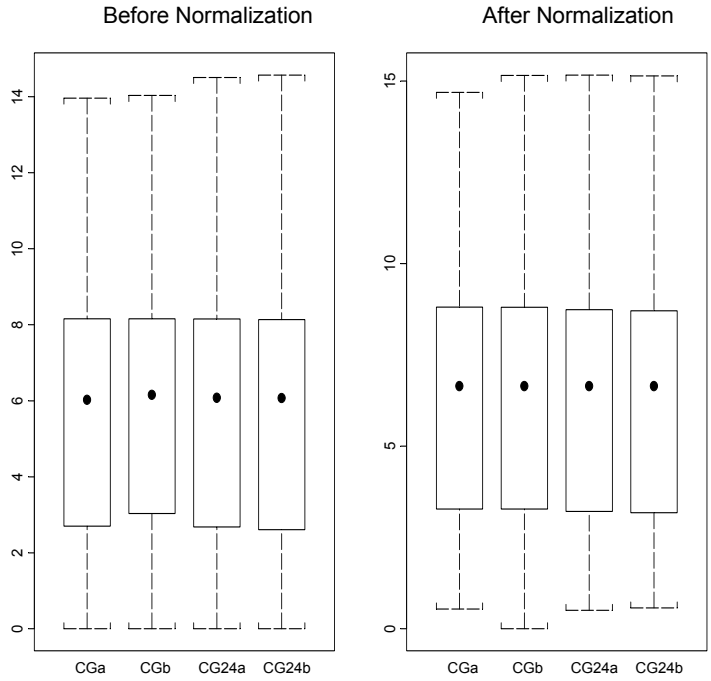


Figure 2.37: *Before and after normalization plots for the Melanoma data logged expression intensities.*

M vs. A Plots

We can do an M vs. A plot of the logged expression intensities in **LCG.N** with the `mva.pairs` function. For MAS4/5 data this function plots all pairwise scatter plots of M vs. A for each treatment condition and replicate combination. Because there are over 12,000 probes on each array we randomly sample 2,000 of them before plotting, and because the intensities have already been logged we turn that off in the plotting function.

```
### First remove missing values
> LCG.N <- na.exclude(LCG.N)
> graphsheet()
> mva.pairs(LCG.N[sample(dim(LCG.N)[1], 2000), ], log = F)
```

The resulting plots are displayed on the following page in Figure 2.38.

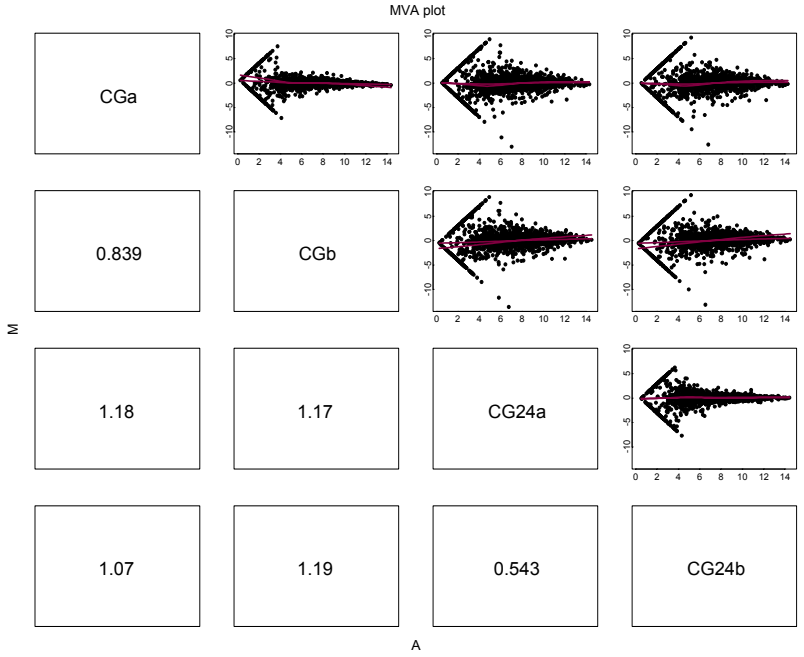


Figure 2.38: *M versus A plots for the Melanoma experiment. For each graph the vertical axis is M and the horizontal axis is A. M is computed from the two arrays found by going horizontally left and vertically down to the first array name you come to. For example, for the upper left scatterplot, just right of the CGa label, M is computed as the difference in (logged) intensities from the CGa and CGb arrays. A is the average of the same (logged) intensities.*

Differential Expression Testing

The cone-shaped MvA plot shows that variance decreases as a function of the log average expression intensity. Given this pattern, we will use the LPE test for differential expression since it allows variance to be modeled as a function of the average expression intensity. We first have to create a couple of objects that are arguments to the LPE test function.

LPE Test

The LPE test function requires baseline variance estimates before computing test statistics. We compute the baseline variance (or error) estimate with the `baseOLIG` function as follows:

```
OLIGgrp0 <- baseOLIG(LCG.N[, 1:2])
```

```
OLIGgrp24 <- baseOLIG(LCG.N[, 3:4])
```

The required argument to `baseOLIG` is the expression intensities for all replicates of a given treatment condition. An optional second argument, `number.bins`, sets the number of the bins for the partition used to compute local error estimates. The default value for `number.bins` is 100 implying that 1% of the intensity values will be in each bin.

We are now ready to compute the LPE test. The function used is `lpeOLIG`, which takes three arguments, the baseline variance objects for each treatment condition and the size of the sample (i.e., number of arrays) for each treatment condition. For the Melanoma data, the call is

```
> LPEObj <- lpeOLIG(OLIGgrp0, OLIGgrp24, sample = c(2,2))
> testStat <- LPEObj@output[, "z.stats"]
```

lpeOLIG computes raw p-values, but you can adjust them to control the family-wise error rate (FWER) or False Discovery Rate (FDR) with the `mt.rawp2adjp` function. `mt.rawp2adjp` takes the vector of raw p-values plus a character string indicating the adjustment procedure. The function call is

```
> adjpObj <- mt.rawp2adjp(LPEObj@pvalue,
      proc = "Bonferroni")
```

Plotting Differential Expression Results

We can plot the results in a Graphlet similar to what is obtained from the GUI. We compute fold change first and then call the Graphlet function.

```
> foldChange <- rowMeans(LCG.N[,1:2])-rowMeans(LCG.N[,3:4])
> procedure <- "Bonferroni"
> LPESumm <-
  lpetest.graphlet(LCG.N, adjpObj@adjp[,2], LPEObj@pvalue,
  adjpObj@index, testStat, foldChange, procedure=procedure,
  chip.name="hgu95a", volcano.plot=T, heatmap.plot=T,
  chromosome.plot=T, html.output=F, variance.plot=T,
  smoother.df=10, trim=5, OLIGgrp1=OLIGgrp0,
  OLIGgrp2=OLIGgrp24, var.xlabs=c("A for cg",
  "A for cg24"), summary.name="LPESumm", open.browser=F)
```

The first six critical arguments to `lpetest.graphlet` are

1. the expression set object (e.g, `LCG.N`)
2. the vector of adjusted p-values

3. the vector of raw p-values
4. the order vector from `mt.rawp2adjp` for sorting from most differentially expressed to least
5. vector of fold change values
6. family wise error rate (default = 0.05)
7. p-value adjustment procedure
8. chip name (e.g., hgu95a)

The rest of the arguments are mostly for controlling output: which plots are created, whether it should be HTML output or not, the smoothing parameters for the loess smoother used in the variance plots, etc. See the help file for `lpetest.graphlet` for more detail.

The resulting plot is a java graphlet in an S-PLUS graphics device. Note that HTML output is turned off (`html.output = F`). The graphlet displayed in S-PLUS and the browser are the same except that points are not linked to annotation databases in the S-PLUS display.

The Output Table

The output of the `lpetest.graphlet` function contains information for annotation and for ranking the genes based on the magnitude of differential expression.

```
> LPESumm[1:10,-c(1,6)]
           GeneIndex foldChange Pvalue AdjPvalue
32314_g_at      1608   3.720782      0         0
  35966_at      2509   5.368228      0         0
37420_i_at      2884   3.129978      0         0
37421_f_at      2885   3.034635      0         0
  37484_at      2948   3.262691      0         0
38228_g_at      3214   3.169818      0         0
  38555_at      3301   4.589305      0         0
38928_r_at      3435   3.207205      0         0
40671_g_at      3980   3.205296      0         0
  40755_at      4064   3.046220      0         0
```

The p-values have been sorted from smallest to largest so printing the first 10 rows prints the 10 most (statistically) differentially expressed genes.

It's worth noting that fold change values less than two in absolute value are significant if their standard errors are relatively small. In this experiment, amongst the 25 most significant genes, there are three with very significant differential expression but with absolute fold change less than two. These genes would have not made the cut using a straight fold-change approach to gene discovery.

REFERENCES

Fox J.W., Dragulev B., Fox N., Mauch C. and Nischt R. (2001). "Identification of ADAM9 in human melanoma: Expression, regulation by matrix and role in cell-cell adhesion." Proceedings of International Protelysis Society Meeting.

Lee, JK and O'Connell M. (2003). "An S-PLUS Library for the Analysis of Differential Expression." To appear in *The Analysis of Gene Expression Data: Methods and Software*. Edited by G Parmigiani, ES Garrett, RA Irizarry and SLZeger. Published by Springer, New York.

EXAMPLES: AFFYMETRIX PROBE-LEVEL DATA

3

Affymetrix Probe-Level Data Analysis Workflow **68**

Two-Sample Design **69**
- Melanoma Data 69
- Importing Data 70
- Normalization 77
- Expression Summaries 79
- Differential Expression Analysis 84
- Annotation 92

Two-Way Design **93**
- Mouse Surgery Data 93
- Importing Data 95
- Quality Control Diagnostics 101
- Normalization 105
- Expression Summaries 106
- Differential Expression Analysis 110
- Gene List Management 117
- Annotation 119

References **123**

AFFYMETRIX PROBE-LEVEL DATA ANALYSIS WORKFLOW

The process of analyzing Affymetrix probe-level gene expression data can be done through the S+ARRAYANALYZER menu. To obtain differential expression information from probe level microarray data, we perform the following basic steps:

1. Import and filter the data.
2. Normalization including
   - Adjustment for background noise,
   - Mismatch correction, and
   - Distribution-based normalization.
3. Summarize.
4. Differential expression analysis.
5. Annotation.

In addition, examining array quality and filtering out bad arrays and genes may be necessary and is typically done between import and normalization. Clustering is also a normal part of gene expression discovery and may be performed between all the major steps of the analysis.

TWO-SAMPLE DESIGN

S+ARRAYANALYZER has an assortment of procedures for doing two-sample differential expression analysis. They include the t-test with or without assuming equal variance for the groups, the Wilcoxon rank-sum test and several permutations tests. In addition the LPE test procedure, which produces improved error estimates when there is little replication in the design, is implemented for two-sample problems. All these procedures are suitable for doing simple comparisons between two groups - treatment versus control, tissue 1 versus tissue 2, etc. For more details about two-sample procedures see Chapter 7, Differential Expression Testing.

Melanoma Data

In this section, we step through the analysis of an experiment using an MM5 melanoma cell-line in which a gel matrix that simulates the *in vivo* cellular condition (and progression) of melanoma was added at 2 hours and 24 hours (Fox et al., 2001). This simple experimental design involves one-factor (matrix condition) at two levels (2 and 24 hours); with expression being measured twice on duplicated arrays for each time point. Condition, replication and file name for each chip are displayed in Table 3.2.

Table 3.1: *Experimental design and file association for the melanoma cancer study.*

Experimental Condition	Repetition	chip label	File Name
2 hours	1	**cg2a**	**cg2a.CEL**
2 hours	2	**cg2b**	**cg2b.CEL**
24 hours	1	**cg24a**	**cg24a.CEL**
24 hours	2	**cg24b**	**cg24b.CEL**

The fundamental question of the study is:

Which genes are active at 24 hours that weren't active at 2 hours.

These *differentially expressed* genes are believed to be relevant to tumor invasion and metastasis.

To answer this question, we need to compare gene expression at the two time points - a two-sample problem.

Importing Data

Start by reading in all the arrays. To import Affymetrix data, from the main S-PLUS menu, select **ArrayAnalyzer ▶ Import Data ▶ From Affymetrix**.

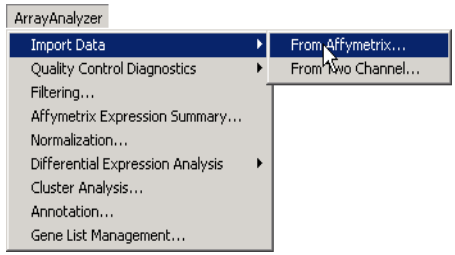


Figure 3.1: *Menu selection to import Affymetrix data.*

Import Affymetrix Data Dialog

Figure 3.25 shows the (**Import Data**) **From Affymetrix** dialog with the **File Selection** page displayed. The primary tasks of the import process are 1) create an experimental design, 2) associate data files with the experimental conditions and 3) save the resulting S-PLUS data object for later use. Secondary tasks include inputting meta (MIAME) data describing the experiment and specifying options for handling data marked as MASKS or OUTLIERS by the Affymetrix software.

The (**Import Data**) **from Affymetrix** dialog has five pages:

1. **File Selection:** This page *must* be completed in order to create a data object for continued analysis.
2. **MIAME:** Completing this page is optional but highly recommended because information on the MIAME tab is used for labeling tables and graphs.
3. **MAS Variables & Filtering:** This page has default settings depending on the type of data files (e.g., MAS4 or MAS5) you select. It also allows the selection of other variables which can be used for more general filtering by using the Filtering dialog.
4. **CEL Filtering:** This page allows a couple of options for filtering out spots not to be used in subsequent analyses when importing probe-level data.

5. **Options:** This page provides options for specifying the number of header lines to skip and the delimiter used in the data file.

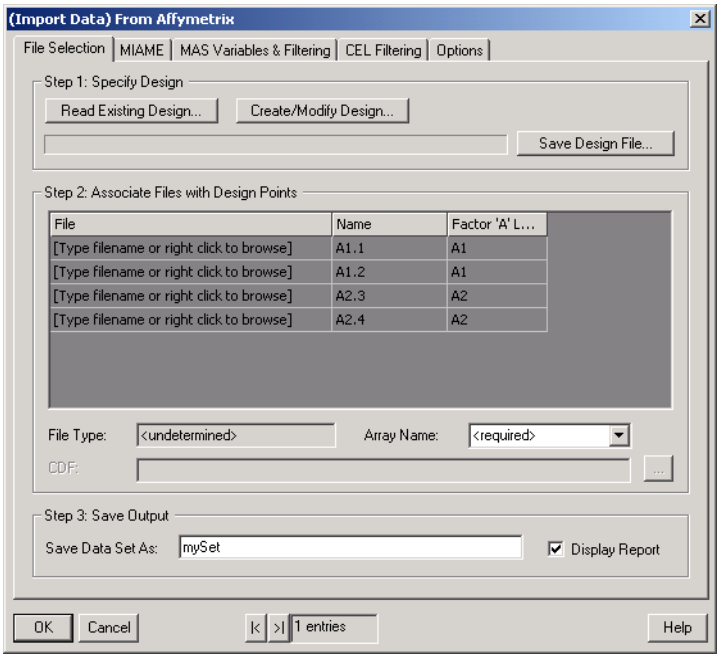


Figure 3.2: *The* ***(Import Data) Affymetrix*** *dialog.*

Step 1: Create The Experiment Design

Before we can begin to associate data files with experimental conditions, we need to set up the experimental conditions in S+ARRAYANALYZER. The easiest way to do this is through the **Create/Modify Design** dialog. Open the **Create/Modify Design** dialog by clicking on the **Create/Modify Design** button on the **File Selection** page of the (**Import Data**) **From Affymetrix** dialog.

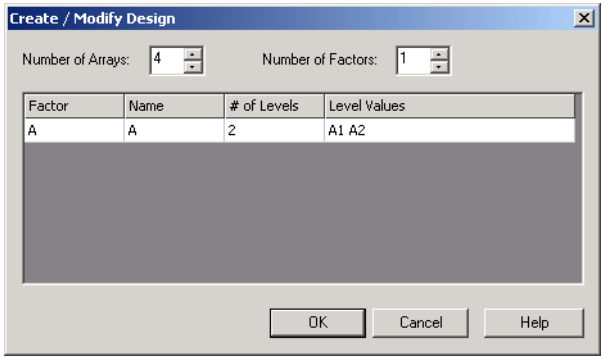


Figure 3.3: *The default **Create/Modify Design** dialog.*

The **Create/Modify Design** dialog allows you to specify

1. The number of arrays to be read.
2. The number of factor in the experiment. Currently one or two are allowed.
3. The name, number of levels and level values for each factor.

To modify the default factor **Name**, **# of Levels** and **Level Values**, type them into the appropriate field. For the two-sample Melanoma analysis, there are 4 arrays and one factor so the first two fields do not change from default. Type in the **Name** of the factor as **Time** and the **Level Values** as **2hr** and **24hr**. The resulting dialog is displayed in Figure 3.4

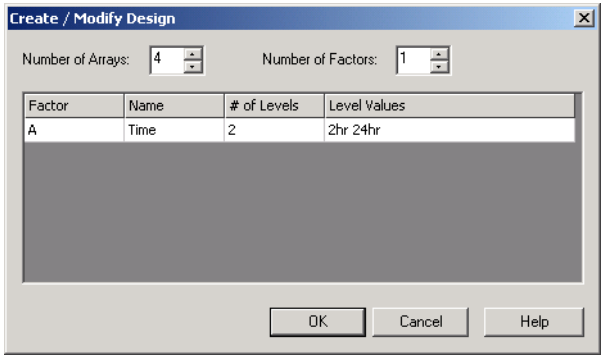


Figure 3.4: *The **Create/Modify Design** dialog with two-sample experiment setup*

Step 2: Associating Files With Design Points

Once the design is complete, click **OK** to copy it into the **File Selection** tab of the (**Import Data**) **From Affymetrix** dialog. Notice that the values for the factor levels have been written into the **Factor** column to facilitate associating files with design points. However, the level values can be reset when the experiment is unbalanced or if you prefer an order different from the default.

The next step is to associate files with each design point. To do so right click in one of the file fields and browse to the location of your files. When you've entered the file names, the dialog should look like Figure 3.5.

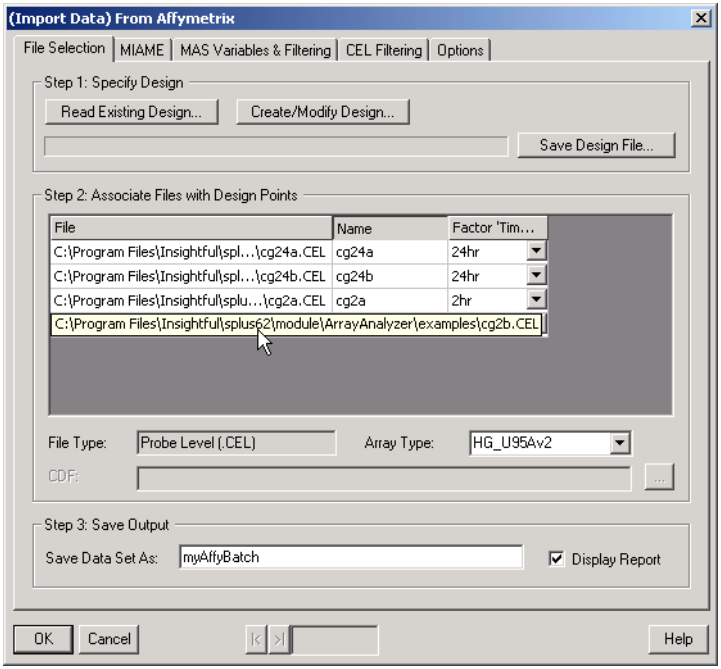


Figure 3.5: *The **(Import Data) From Affymetrix** dialog, after the data files have been entered. To see the full path of a file, hover your mouse over the filename.*

You can find the Melanoma example data by navigating to your **splus62/module/ArrayAnalyzer/examples** directory and selecting the **cg2a.CEL** file. Press the CTRL key and select (in order) the **cg2b.CEL**, **cg24a.CEL**, and **cg24b.CEL**. Alternatively, you can read the design file named **cgDesignFile.txt** in the **examples** directory to load the design and create file associations.

File Type

Note that the **File Type**, **Probe Level** (**.CEL**), listed below the files is automatically detected once a file is selected. The dialog is designed to prohibit mixing file types.

Array Type

The **Array Type** is automatically detected for probe-level data. For this experiment it is **HG_U95Av2** or **hgu95av2** as it is listed in the **Array Name** drop-down list; select **hgu95av2** and continue to the next step, but do not click **OK** until the following steps have been completed.

Step 3: Saving the Data Object

To save the data object, type a name in the **Save As** field near the bottom of the dialog (**Step 3: Save Output**). Remember this name, as it is used in the other analysis steps such as quality checks, filtering and normalization. For our example, enter **cgAffyBatch** as the object name. The **Display Report** checkbox indicates whether or not to print summary information into an S-PLUS report window.

Figure 3.6: *Saving the imported data as **cgAffyBatch**.*

Saving the Design

Once you've entered all the information on this tab you can save it for later use by clicking the **Save Design** button at the top of the dialog. A **.txt** file is written to the directory of your choice with the number of arrays, number of factors, number of levels, and the full path file names and their associated factor levels.

Reading Designs

This *design* file can be reused for another experiment with the same design by modifying the file locations and names and factor levels as needed. In fact, if you have many arrays in your experiment you can create a file with all the design content and read it with the **Read Design** button located at the top of the dialog. Reading the design in

this way will fill the file name fields and their associated factor levels. For information about creating a design file see Appendix A, Creating a Design File.

MIAME Page

MIAME is an acronym for Minimal Information About a Microarray Experiment. This information should be entered on the second page of the (**Import Data**) **from Affymetrix** dialog. This information is not required, but it is used to label the resulting S-PLUS object for later identification. Once you've entered MIAME information for any experiment, the first three fields are saved and are filled automatically the next time you open this dialog.

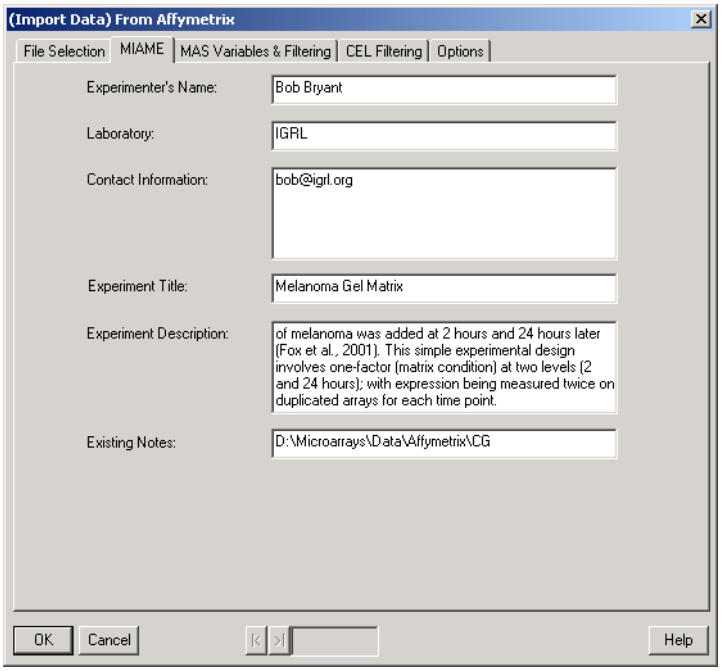


Figure 3.7: *Entering experiment information on the **MIAME** page.*

MAS Variables & Filtering Page

The third page of the (**Import Data**) **From Affymetrix** dialog is for variable and row selection for Affymetrix MAS 4/5 data. It is not used for **CEL** data.

CEL Filtering

The fourth page of the (**Import Data**) **From Affymetrix** dialog is for spot filtering probe-level (CEL) data. The options are:

1. A checkbox to convert spots labeled as **MASKS** to missing so they aren't used in subsequent analyses.
2. A checkbox to convert spots marked as **OUTLIERS** to missing so they aren't used in subsequent analyses.

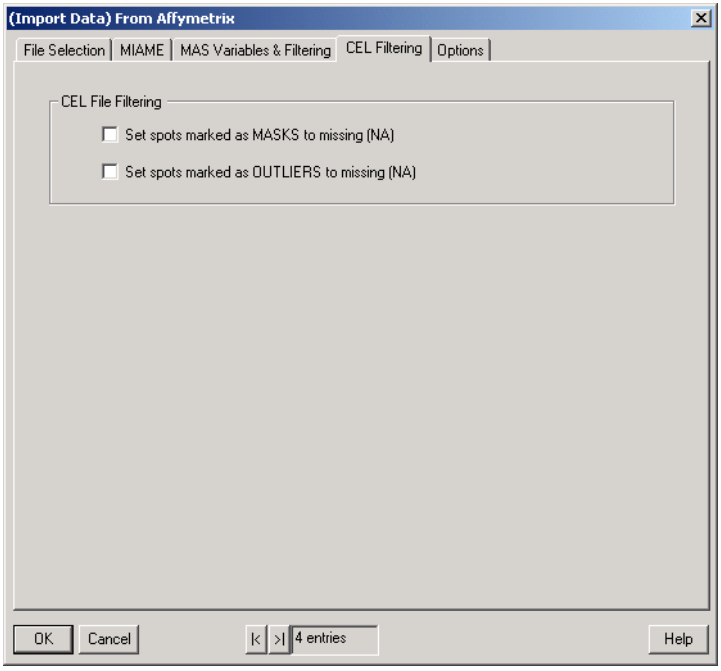


Figure 3.8: ***CEL Filtering*** *page of the* ***(Import Data) From Affymetrix*** *dialog.*

Options

The last page on the (**Import Data**) **From Affymetrix** dialog is the **Options** tab. The tab provides two options used during data import:

1. The number of header lines to skip in each file before reading the data. Normally this can be detected automatically but it is provided as an option for unusual cases where auto detection can not find the row with column names.
2. The delimiter separating the fields in each line of the data files. Normally this can be detected automatically but it is provided as an option for unusual cases where auto detection can not determine the field delimiter.

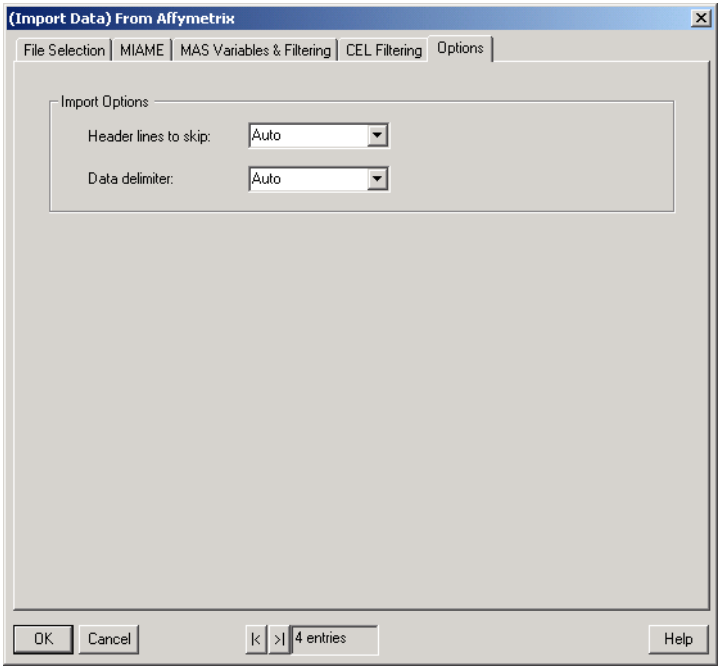


Figure 3.9: ***Options*** *page of the* ***(Import Data) From Affymetrix*** *dialog.*

Press **OK** when you have completed the dialog and the data are imported. It is now ready for use in S+ARRAYANALYZER.

Normalization

Normalization procedures may be applied to both raw probe-set intensities *and* to summarized expression intensities. For examples of normalizing expression summary data see Chapter 2, Examples: Affymetrix MAS Data and Chapter 6, Pre-Processing and Normalization.

In this section we focus on normalizing probe-set data without summarizing it first.

Normalization Dialog

Open the **Normalization** dialog by selecting **Normalization** from the **ArrayAnalyzer** drop-down menu.

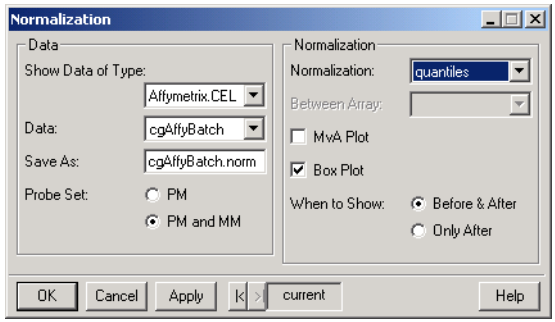


Figure 3.10: *The **Normalization** dialog.*

Select **Affymetrix CEL** from the **Show Data of Type** drop-down list and choose the **cgAffyBatch** data object. Explore the normalization procedures available in the **Normalization** drop-down list; select the **quantiles** procedure to normalize both PM (perfect match) and MM (mismatch) intensities or the PM intensities only. For this example, select **PM and MM** to normalize both.

Save As

The **Save As** field takes an object name for saving the normalized **affyBatch** (probe-level) object with the default name **cgAffyBatch.norm**.

Clicking **OK** creates the normalized **affyBatch** object and plots pre- and post-normalization boxplots for comparison. The plot is on a $\log_2$ scale, but the expression intensities are saved on the original (raw) intensity scale.

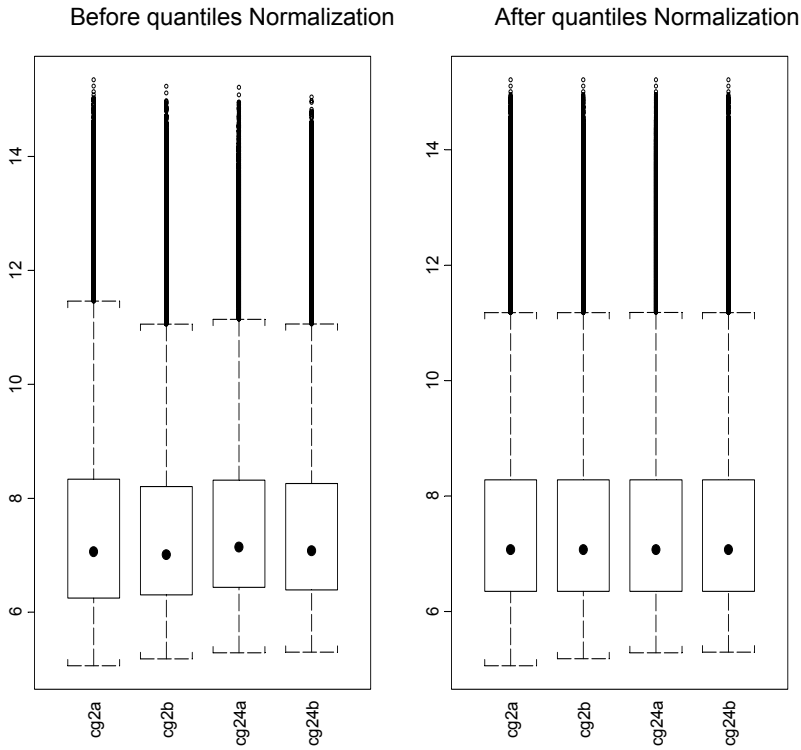


Figure 3.11: *Pre- and post-normalized probe-level data. The normalized intensities are plotted on the log_2 scale.*

Expression Summaries

Once we've imported the data files we need to convert the raw probe-level expression intensities to expression *summaries* before testing for differential expression. This is usually done in a series of steps including some combination of the following:

- Background correction.
- Normalization.
- Probe-specific background correction, e.g., subtracting mismatch (MM) expression intensities.
- Summarizing the probe set values into one expression measure and sometimes a standard error for this measure.

An assortment of procedures are available for completing these steps. You can find much more detail in Chapter 6, Pre-Processing and Normalization. In addition to normalization in the context of summarizing raw intensities you can also normalize without the summarization step.

RMA Summary

In this section, we focus on one sequence of steps referred to as robust multichip analysis or RMA for short. This procedure completes the following steps:

1. Probe specific correction of the perfect-match (PM) probes using a model based on observed intensity being the sum of signal and (background) noise (Irizarry et. al., 2002; Irizary et. al., 2003),
2. Normalization of corrected PM probes using quantile normalization (Bolstad et. al., 2002),
3. Calculation of expression measures using median polish.

This sequence of steps is available by simply checking the **RMA** radio button in the upper right corner of the **Affymetrix Expression Summary** dialog. Open the dialog by clicking **ArrayAnalyzer ► Affymetrix Expression Summary** from the main S-PLUS menu bar. Then select the **cgAffyBatch** object in the **CEL Data** drop-down list, and select the **RMA** checkbox. The result of the computation is an expression summary object. Set the name to be **cgExprSet.rma** by typing it into the **Save As** field. Figure 3.12 displays the **Affymetrix Expression Summary** dialog ready to go.

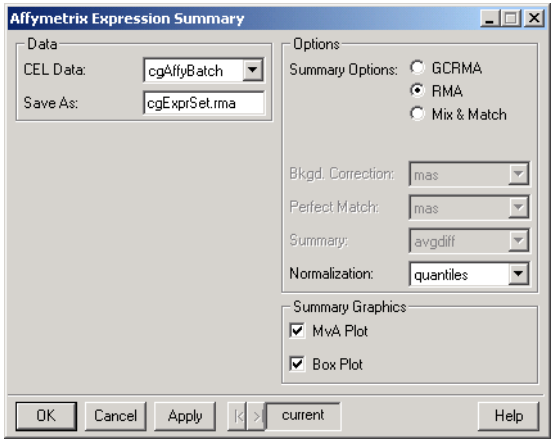


Figure 3.12: *Specifying Robust Multichip Analysis with a single checkbox.*

RMA Output

A sequence of graphs is produced as output by the RMA procedure. Figure 3.13 displays the MvA plot (the expression intensity log ratio, M, vs. the overall average intensity, A) for the two samples taken at two (2) hours. The value in the lower left panel is the inter-quartile range (IQR) of the values of M across all (summarized) expression values. A small value indicates there is little difference (on the $\log_2$ scale) for the middle 50% of the expression values for the two chips. For replicate chips, there is no *real* differential expression so the IQR is expected to be small.

Figure 3.14 displays the MvA plot for the 24 hour samples. The interpretation is the same as that for Figure 3.13.

Figure 3.15 displays boxplots of logged expression summaries for each sample chip. Visual inspection shows the distributions are well aligned at their centers and quartiles. Although normalization may be repeated sequentially to summarized expression intensities there is little need to apply more normalization to `cgExprSet.rma`.

Note

The values displayed in the MvA plots in Figures 3.13 and 3.14 depend on the values used for `.Random.seed`. You may see slightly different plots as a result.

Log_2 of Expression Intensities

After applying normalization and summarization procedures to the raw expression intensities, a $\log_2$ transformation is applied. Consequently, the returned summarized object contains expression intensities on a log scale. The log transformation is computed as

$\log_2(\text{E})$ if $\text{E} > 1$

0 if E less than or equal to 1.

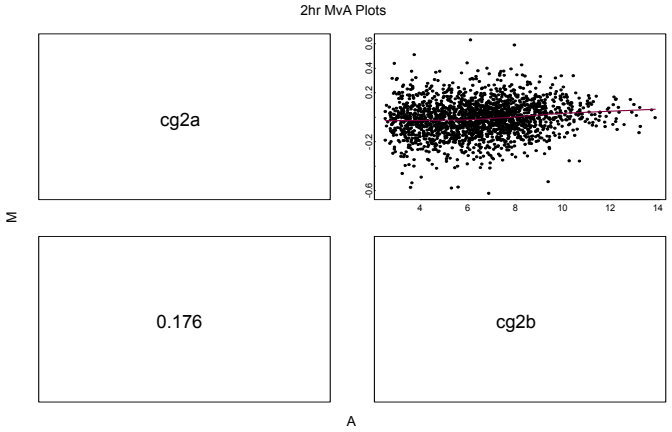


Figure 3.13: *MvA plot for the two replicate samples measured at two hours. The value in the lower left panel of the plot is the interquartile range of M.*

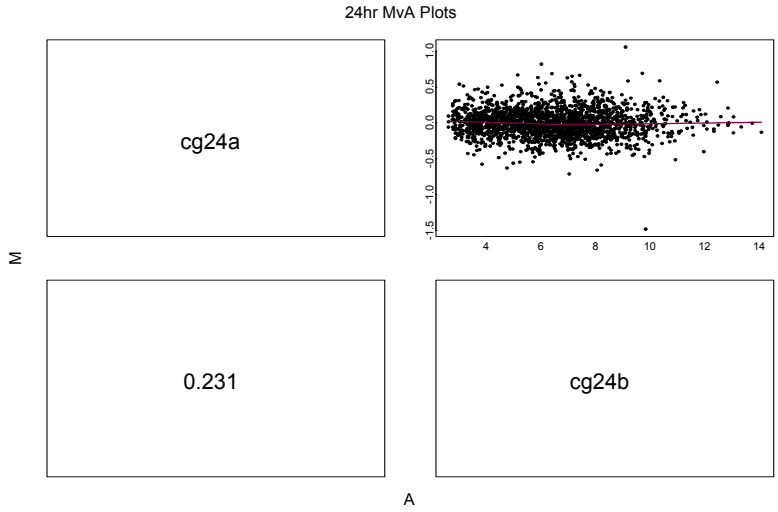


Figure 3.14: *MvA plot for the two replicate samples measured at 24 hours. The value in the lower left panel of the plot is the interquartile range of M.*

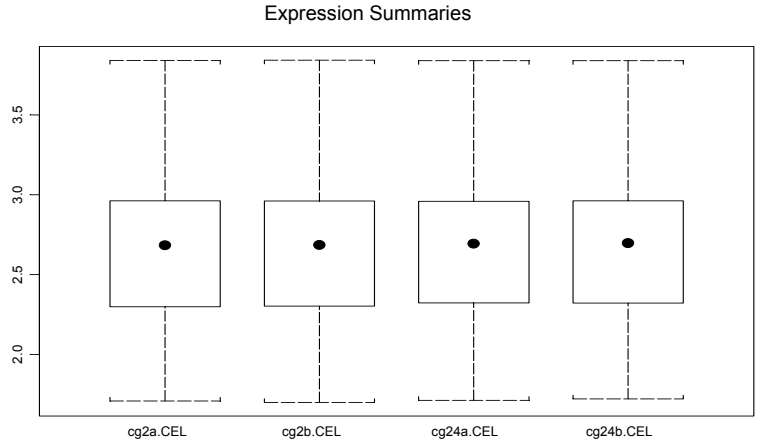


Figure 3.15: *Boxplot of log_2 expression intensities for the four samples after applying the composite RMA procedure.*

Differential Expression Analysis

After normalizing and summarizing the probe-level data we are ready to do differential expression analysis. From the main menu, open the **Local Pooled Error Test** dialog by clicking **ArrayAnalyzer ► Differential Expression Analysis ► LPE Test**.

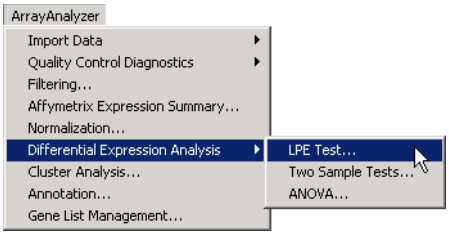


Figure 3.16: *Selecting **LPE Test** to open the **Local Pooled Error Test** dialog.*

Local Pooled Error Test

The Local Pooled Error (LPE) Test dialog contains groups for specifying the data and levels of the factor you wish to compare, how to adjust for multiple testing, options for controlling variance function estimation and various output options.

Options

Other than selecting the data and factor levels for comparison, the most critical inputs are in the Options group. The **Options** group allows you to set the family-wise error rate (FWER) or the false discovery rate (FDR) to control the overall Type I error rate (false positive rate) based on adjusting individual test p-values to account for multiple tests. In our melanoma example there are 12,558 genes so the increase in Type I error is substantial if you don't adjust the p-values.

There are many options for adjusting the p-values to achieve the FWER (or FDR). We describe them in more detail in Chapter 7, Differential Expression Testing. Here we leave the default setting as **Bonferroni**.

Output Options

There are five output options in the **Output Options** group:

1. Volcano plot.
2. Heat map.
3. Chromosome plot.

4. Variance plots.
5. Top 15 genes list.

Setting Up the Dialog

To set up the **Local Pooled Error Test** dialog, follow these steps:

1. In the **Show Data of Type** field, select **Affymetrix**.
2. In the **Data** field, select **cgExprSet.rma**.

   Select the **Factor**, **Compare Level 1** and **Compare Level 2** fields to be **Time**, **2hr** and **24hr**, respectively. The **Array Name** field should be automatically filled with **hgu95av2**.
3. Click the drop-down arrow of the **Adjustment** field in the **Options** group to set the **FWER/FDR** procedure to **Bonferroni**.
4. Ensure that all the **Output Options** are checked. These include:
   - Volcano plot.
   - Heat map.
   - Chromosome plot.
   - Variance plots.
   - Top 15 genes table.
5. In the **Output** group:
   - Deselect the **Display Output in S-PLUS** checkbox.
   - Select the **Save Output as HTML** checkbox.
   - Select the **Display HTML Output** checkbox, and enter **cgLPEBH.html** for the filename.
   - Enter **cgLPEBon** in the **Save As** field in the lower right corner of the dialog for saving the test result object.

The resulting dialog is displayed in Figure 3.17.

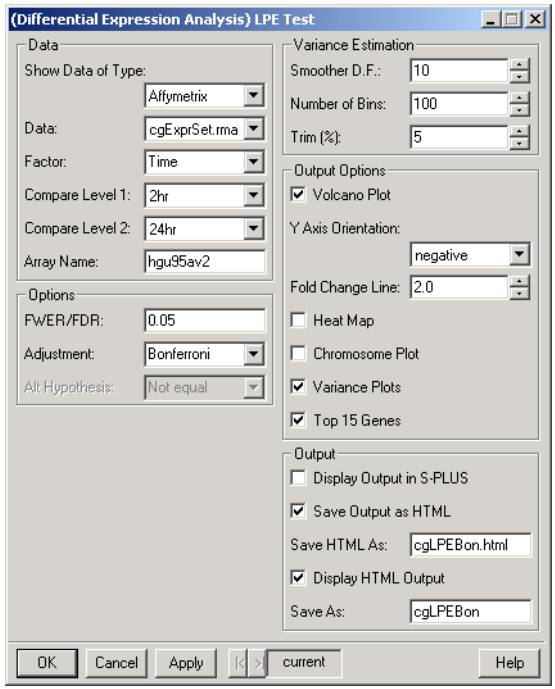


Figure 3.17: *The **(Differential Expression Analysis) LPE Test** dialog ready to run the tests for differential expression for the melanoma data.*

Top 15 Genes Summary Table

The first page of the output is the Top 15 Gene list. Included in the table is the test statistic, the raw p-value, the adjusted p-value and the fold change. When the output is displayed in HTML and annotation data is available for the chip, each gene identifier at the beginning of each row of the table is hyperlinked to one or more annotation database.

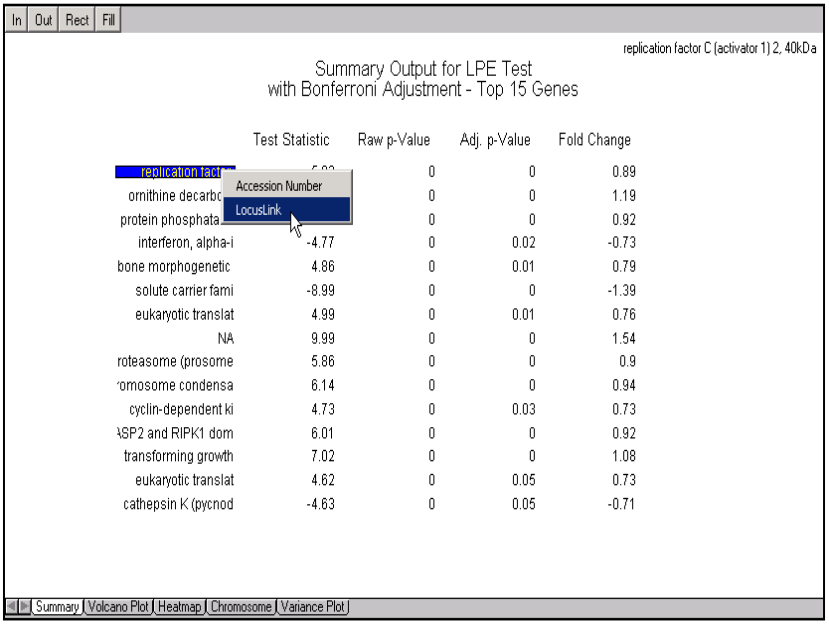


Figure 3.18: *Summary of top 15 differentially expressed genes. Each gene is hyperlinked to annotation databases.*

Volcano Plot

A *volcano* plot displays the logarithm of adjusted p-value versus fold change, as shown in Figure 3.19. The vertical lines indicate fold change values of plus or minus one, and the horizontal line indicates a significant test p-value after doing the Bonferroni correction. Points located in the upper, outer sextants are those with large absolute fold change and small (significant) p-value. Each of those points is active so you can click on a point to access annotation information from Locus Link or GenBank.

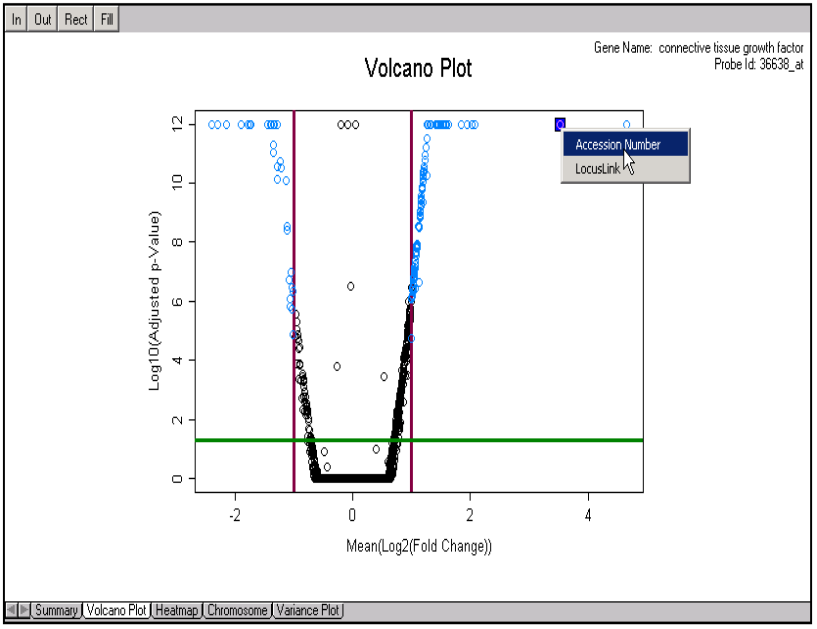


Figure 3.19: *A volcano plot, which is the logarithm of p-value versus fold change. Points below the horizontal line are hyperlinked to annotation databases.*

Heat Map Plot

A *heat map* plot, shown in Figure 3.20, shows a two-way layout of the most differentially expressed genes along the vertical axis versus the experimental conditions on the horizontal axis. This graph is also hyperlinked to the annotation information.

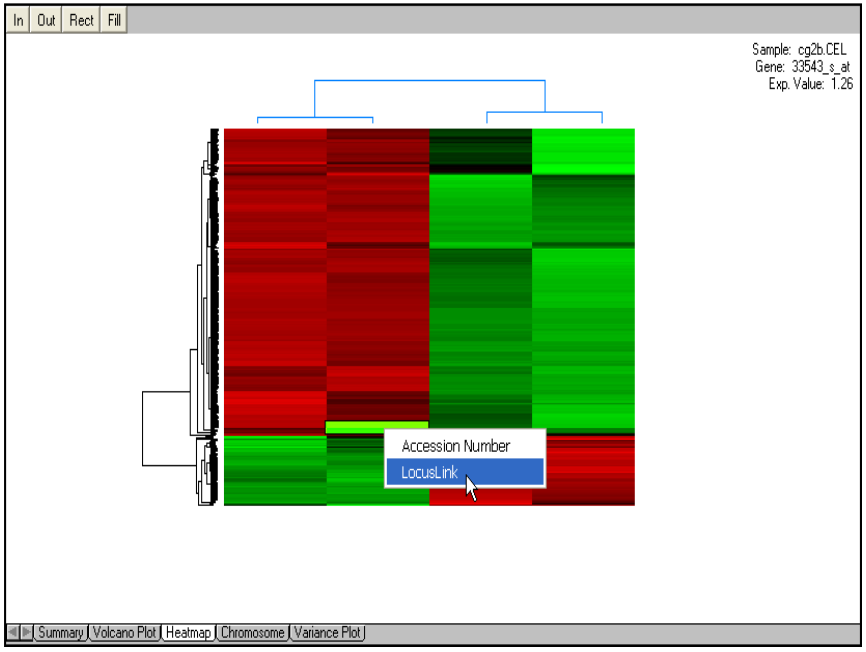


Figure 3.20: *A heat map plot shows differentially expressed genes as a function of experimental conditions. The map is hyperlinked to annotation databases.*

Chromosome Plot A *chromosome* plot displays the entire chromosome with differential expression marked (up for positive, down for negative) for each gene represented on the chip. The top 15 differentially expressed genes are highlighted with color (orange) to indicate their location on the chromosome. Hovering the mouse over one of the colored (active) points displays the gene ID in the upper right-hand corner of the graph, as shown in Figure 3.21.

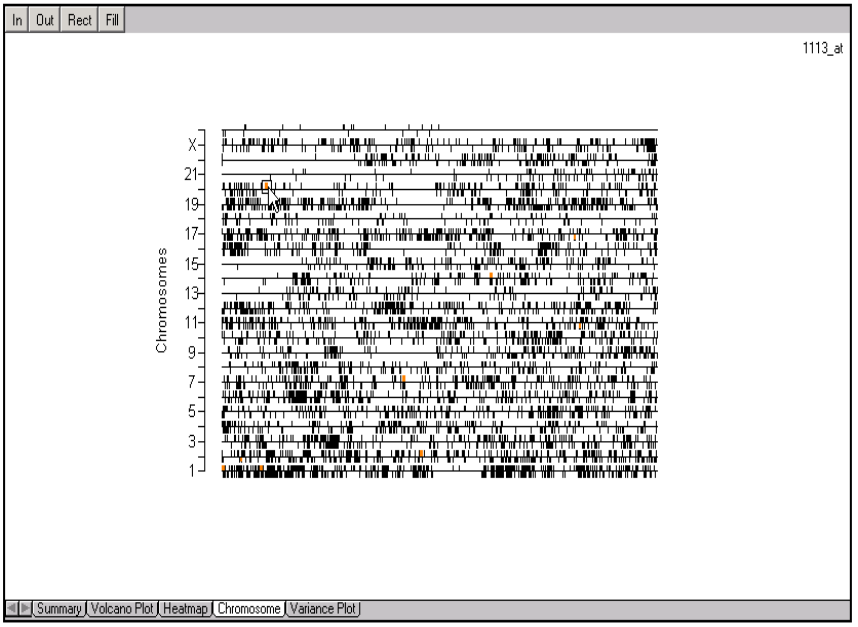


Figure 3.21: *The chromosome plot displays the entire chromosome with differential expression marked (up for positive, down for negative) for each gene on the chip. The orange color indicates the location of the top 15 most significant differentially expressed genes.*

Variance Plots

The *variance* plots display the variance estimates used for the LPE test as a function of differential expression for each treatment condition. In this example, the plot shows the variance decreasing dramatically as differential expression increases, as shown in Figure 3.22.

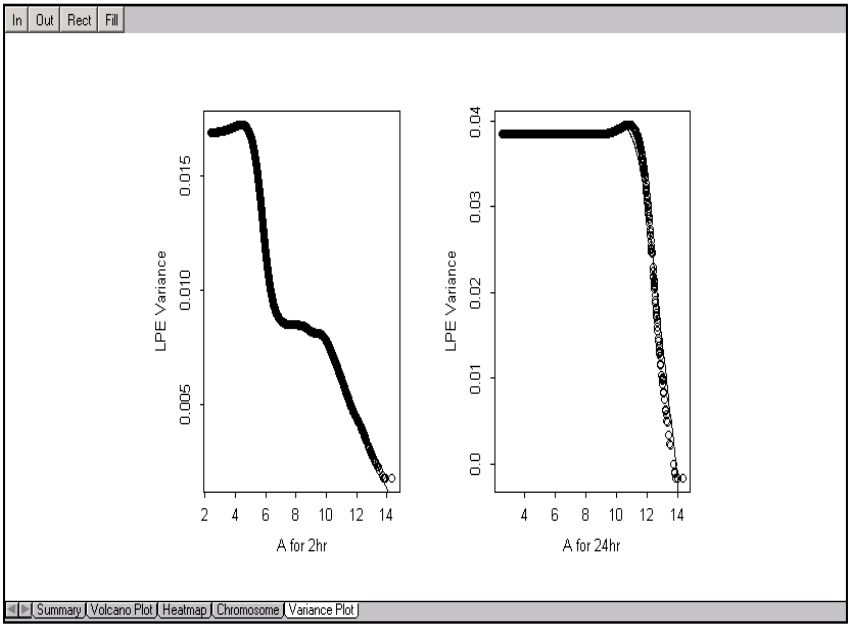


Figure 3.22: *Variance plots for the 2-hour and 24-hour data.*

Annotation

Clicking one of the hyperlinked points in one of the Top 15 Genes Summary, the volcano plot, or the heat map pops up a menu for selecting the database to query for annotation information. Selecting either one opens an HTML page in your default web browser displaying a brief description of the gene with hyperlinks to more detailed information. Figure 3.23 shows an example page from LocusLink with annotation for one of the differentially expressed genes in the melanoma example.

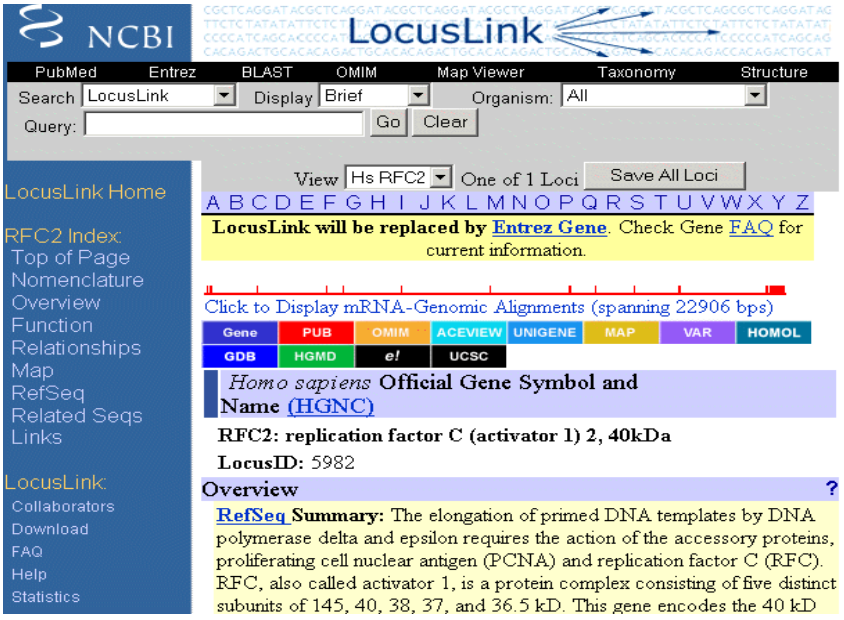

NCBI
LocusLink
PubMed Entrez BLAST OMIM Map Viewer Taxonomy Structure
Search LocusLink Display Brief Organism: All
Query: Go Clear
LocusLink Home
RFC2 Index:
Top of Page
Nomenclature
Overview
Function
Relationships
Map
RefSeq
Related Seqs
Links
LocusLink:
Collaborators
Download
FAQ
Help
Statistics
View Hs RFC2 One of 1 Loci Save All Loci
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
LocusLink will be replaced by Entrez Gene. Check Gene FAQ for current information.
Click to Display mRNA-Genomic Alignments (spanning 22906 bps)
Gene PUB OMIM ACEVIEW UNIGENE MAP VAR HOMOL
GDB HGMD e! UCSC
Homo sapiens Official Gene Symbol and Name (HGNC)
RFC2: replication factor C (activator 1) 2, 40kDa
LocusID: 5982
Overview
RefSeq Summary: The elongation of primed DNA templates by DNA polymerase delta and epsilon requires the action of the accessory proteins, proliferating cell nuclear antigen (PCNA) and replication factor C (RFC). RFC, also called activator 1, is a protein complex consisting of five distinct subunits of 145, 40, 38, 37, and 36.5 kD. This gene encodes the 40 kD

Figure 3.23: *Annotation information from LocusLink.*

TWO-WAY DESIGN

In addition to the two-sample procedures described briefly at the beginning of this chapter, S+ARRAYANALYZER provides ANOVA methods for analyzing one- and two-way designs as well. The statistical methodology for two-sample and multi-sample one- and two-way designs is discussed in detail in Chapter 7, Differential Expression Testing. In this section we focus on a single example, a two-way design to study the differences in healing processes for young and old mice after an injury.

Mouse Surgery Data

In this section, we step through the analysis of an experiment using mice with surgery-induced injuries to discover genes active in the healing process. The experimental design involves two factors: 1) the *age* of the mice recorded as young or old, and 2) the time of observation taken at 0, 0.5, 1, 2, and 4 hours following the surgery. Gene expression was measured in three-replicate sets (i.e., three mice) for each time point and each age combination. The main hypothesis of interest involves discovering genes showing differential expression between the two age groups and over time.

To demonstrate S+ARRAYANALYZER we focus on three time points - 0, 1 and 4 hours. The arrays and data files are listed in Table 3.2.

Table 3.2: *Experimental design and file association for the mouse surgery study.*

Age	Time	Rep	Array label	File name
Old	0hr	1	Old0hr1	Old0hr1
Old	0hr	2	Old0hr2	Old0hr2
Old	0hr	3	Old0hr3	Old0hr3
Old	1hr	1	Old1hr1	Old1hr1
Old	1hr	2	Old1hr2	Old1hr2
Old	1hr	3	Old1hr3	Old1hr3
Old	4hr	1	Old4hr1	Old4hr1

Table 3.2: *Experimental design and file association for the mouse surgery study.*

Age	Time	Rep	Array label	File name
Old	4hr	2	Old4hr2	Old4hr2
Old	4hr	3	Old4hr3	Old4hr3
Young	0hr	1	Young0hr1	Young0hr1
Young	0hr	2	Young0hr2	Young0hr2
Young	0hr	3	Young0hr3	Young0hr3
Young	1hr	1	Young1hr1	Young1hr1
Young	1hr	2	Young1hr2	Young1hr2
Young	1hr	3	Young1hr3	Young1hr3
Young	4hr	1	Young4hr1	Young4hr1
Young	4hr	2	Young4hr2	Young4hr2
Young	4hr	3	Young4hr3	Young4hr3

Setting Up The Analysis

The fundamental question posed by this study is:

> *Are there any differences in gene expression between the young and old age groups?*

A secondary question is:

> *Do the differentially expressing genes change over time?*

The Two-Way Design

To answer these questions, the example focuses on data for the young and old mice collected at 0 hours, 1 hour and 4 hours. The experimental design is a balanced two-way design as displayed in Table 3.3.

Table 3.3: *Experimental conditions for the examining differential expression in healing for young versus old mice.*

Young	Old
3 reps @ 0 hours	3 reps @ 0 hours
3 reps @ 1 hour	3 reps @ 1 hour
3 reps @ 4 hours	3 reps @ 4 hours

Importing Data

Start by reading in all the arrays. To import Affymetrix data, from the main S-PLUS menu, select **ArrayAnalyzer ▶ Import Data ▶ From Affymetrix**.

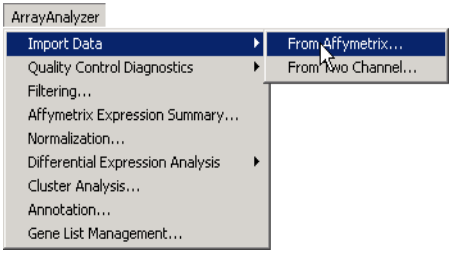


Figure 3.24: *Menu selection to import Affymetrix data.*

Import Affymetrix Data Dialog

Figure 3.25 shows the **Import Affymetrix Data** dialog with the **File Selection** page displayed. The primary tasks of the import process are 1) create an experimental design, 2) associate data files with the experimental conditions and 3) save the resulting S-PLUS data object

for later use. Secondary tasks include inputting meta (MIAME) data describing the experiment and specifying options for handling data marked as MASKS or OUTLIERS.

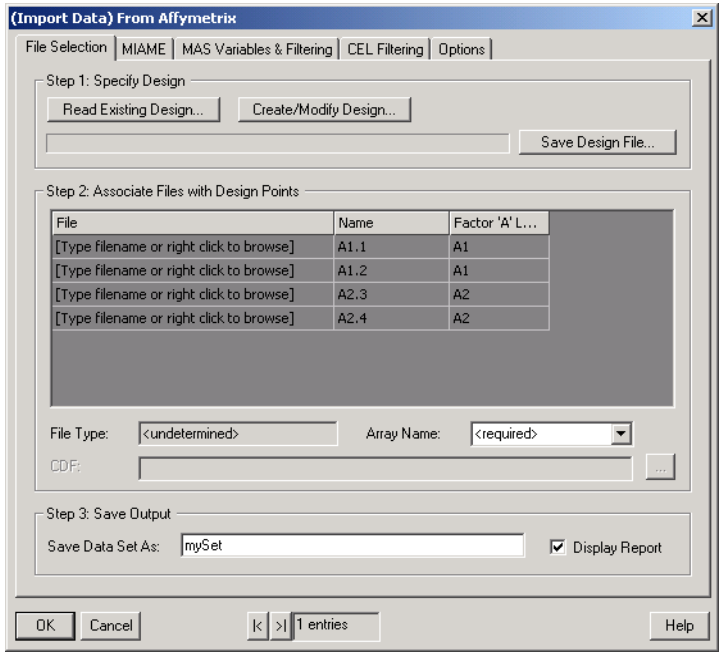


Figure 3.25: *The **Import Affymetrix Data** dialog.*

The (**Import Data**) **from Affymetrix** dialog has five pages:

- **File Selection** This page *must* be completed in order to create a data object for continued analysis.
- **MIAME** Completing this page is optional but highly recommended because information on the **MIAME** tab is used for labeling tables and graphs.
- **MAS Variables & Filtering** This page has default settings depending on the type of data files (e.g., MAS4 or MAS5) you select. It also allows the selection of other variables which can be used for more general filtering by using the **Filtering** dialog.
- **CEL Filtering** This page allows a couple of options for filtering out spots not to be used in subsequent analyses when importing probe-level data.
- **Options** This page provides options for specifying the number of header lines to skip and the delimiter used in the data file.

Step 1: Create The Experiment Design

Before we can begin to associate data files with experimental conditions, we need to set up the experimental conditions in S+ARRAYANALYZER. The easiest way to do this is through the **Create/Modify Design** dialog. Open the **Create/Modify Design** dialog by clicking on the **Create/Modify Design** button on the **File Selection** page of the (**Import Data**) **From Affymetrix** dialog

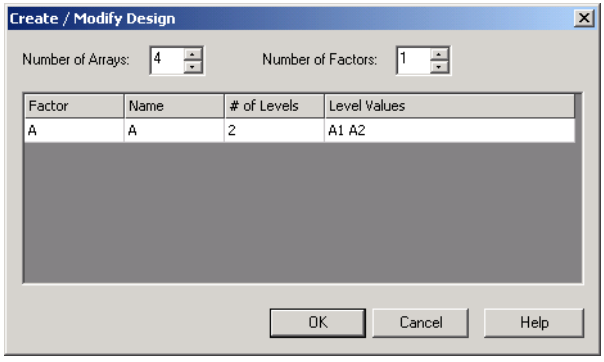


Figure 3.26: *The default* ***Create/Modify Design*** *dialog.*

The **Create/Modify Design** dialog allows you to specify

1. The number of arrays to be read.
2. The number of factors in the experiment. Currently one or two are allowed.
3. The name, number of levels and level values for each factor.

To modify the default factor **Name**, **# of Levels** and **Level Values**, type them into the appropriate field. For the two-way analysis of the Surgery data, there are 18 arrays and two factors. The Age factor has two levels - Young and Old. The Time factor has three levels - 0hr, 1hr and 4hr. The resulting dialog is displayed in Figure 3.27

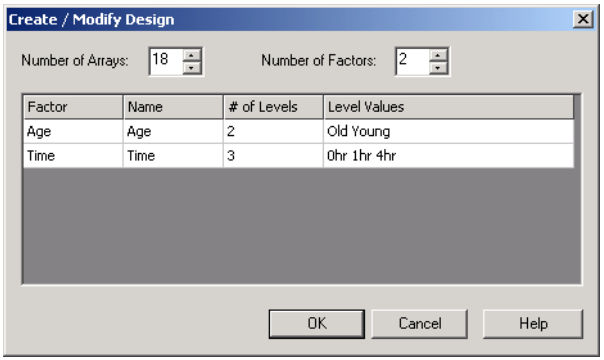


Figure 3.27: *The **Create/Modify Design** dialog with two-way experiment setup*

Step 2: Associating Files With Design Points

Once the design is complete, click **OK** to copy it into the **File Selection** page of the (**Import Data**) **From Affymetrix** dialog. Notice that the number of rows for the **File Selection** box is modified to match the number of arrays specified on the **Create/Modify Design** dialog. Furthermore, values for the factor levels have been written into the **Factor** columns to facilitate associating files with design points. If the experiment is balanced the factor level settings will be exactly as needed. However, the level values can be reset when the experiment is unbalanced or if you prefer an order different from the default.

The next step is to associate files with each design point. To do so right click in one of the file fields and browse to the location of your files.

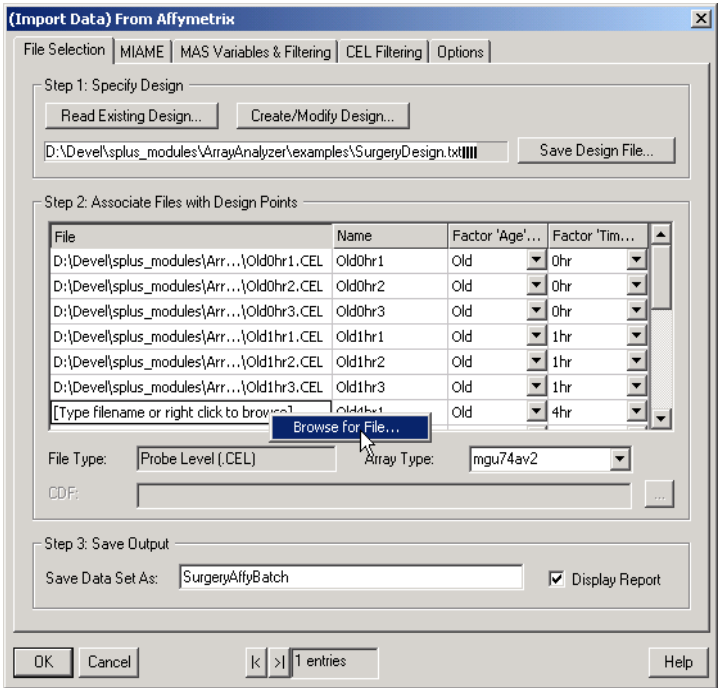


Figure 3.28: *Browsing for data files.*

You can find the surgery example data by navigating to your **splus62/module/ArrayAnalyzer/examples** directory and selecting the **Old0hr1.txt** file. Repeat for the other 17**.txt** files, entering one file per field. Alternatively, you can read the design file named **SurgeryDesign.txt** in the examples directory to load the design and create file associations.

File Type

Note that the **File Type**, **Probe Level** (**.CEL**), listed below the files is automatically detected once a file is selected. The dialog is designed to prohibit mixing file types.

The Array Type

The **Array Type** is automatically detected for probe-level data. For this experiment it is **MG_U74Av2** or **mgu74av2** as it is listed in the **Array Type** drop-down list.

Step 3: Saving the Data Object

To save the data object, type a name in the **Save As** field near the bottom of the dialog (**Step 3: Save Output**). Remember this name, as it is used in the other analysis steps such as quality checks, filtering and normalization. For our example, enter **SurgeryAffyBatch** as the object name. The **Display Report** checkbox indicates whether or not to print summary information into an S-PLUS report window.

Figure 3.29: *Saving the imported data as* ***SurgeryAffyBatch****.*

Saving the Design

Once you've entered all the information on this tab you can save it for later use by clicking the **Save Design** button at the top of the dialog. A **.txt** file is written to the directory of your choice with number of factors, number of levels, repetitions and the full path file names and their associated factor levels.

Reading Designs

This *design* file can be reused for another experiment with the same design by modifying the file locations and names and factor levels as needed. In fact, if you have many arrays in your experiment you can create a file with all the design content and read it with the **Read Design** button which will fill the file name fields and their associated factor levels.

MIAME Page

MIAME is an acronym for Minimal Information About a Microarray Experiment, and this information can be entered on the second page of the (**Import Data**) **from Affymetrix** dialog. This information is not required, but it is used in table output and graphics, and thus it is to your advantage to complete the information in this page. Once you've entered MIAME information for any experiment, the first three fields are saved and are filled automatically the next time you open this dialog.

MAS Variables & Filtering Page

The third page of the (**Import Data**) **From Affymetrix** dialog is for variable and row selection for Affymetrix MAS 4/5 data. It is not used for **CEL** data.

CEL Filtering

The fourth page of the (**Import Data**) **From Affymetrix** dialog is for spot filtering probe-level data. The options are:

1. A checkbox to convert spots labeled as MASKS to missing so they aren't used in subsequent analyses.
2. A checkbox to convert spots marked as OUTLIERS to missing so they aren't used in subsequent analyses.

Other Options

The last page on the (**Import Data**) **From Affymetrix** dialog is the **Options** tab. The tab provides two options used during data import:

1. The number of header lines to skip in each file before reading the data. Normally this can be detected automatically but it is provided as an option for unusual cases where auto detection can not find the row with column names.
2. The delimiter separating the fields in each line of the data files. Normally this can be detected automatically but it is provided as an option for unusual cases where auto detection can not determine the field delimiter.

Press **OK** when you have completed the dialog and the data are imported. It is now ready for use in S+ARRAYANALYZER.

Quality Control Diagnostics

Once the data is imported, open the **Quality Control Diagnostics** dialog to do quality checks. Open the dialog by clicking through the sequence **ArrayAnalyzer ▶ Quality Control Diagnostics ▶ Affymetrix** from the main S-PLUS menu bar.

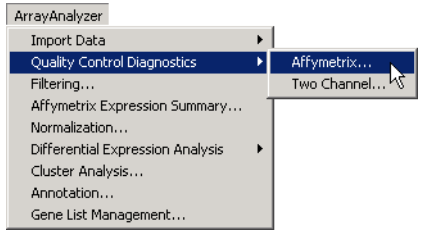


Figure 3.30: *Opening the **Quality Control Diagnostics** dialog.*

The resulting dialog provides a number of options for doing visual quality checks.

1. **Image Plot:** An image plot of each array. For Affy CEL data each pixel corresponds to a different spot on the array.

2. **MvA Plot:** of replicates versus each other within treatment condition or versus the median expression computed for all replicates within each treatment condition. Options for the MvA plot are a) a scatter plot produced by drawing a random sample of genes and b) a hexbin plot of all genes where the hexagonal "points" are colored to give a sense of the density of points at each location.
3. **Genes Present Plot:** A simple barplot of percent of genes present on each array. Available only for MAS 4/5 data when the Detection column has been selected for filtering during the data import stage.
4. **Intensity Boxplot:** Boxplots of expression intensities for each array.
5. **RNA Degrad. Plot:** Plot of RNA degradation. Only available for probe-level (CEL) data.
6. **Principal Components Plot:** Plot of the first two principal components using treatment combinations (i.e., expression intensities for the entire array) as variables in the principal components analysis.

For a thorough analysis we would typically create most if not all of the diagnostic plots and check them for problem arrays and spots. For this example we will do a few CEL-specific plots as examples and refer you to the other chapters Chapter 2, Examples: Affymetrix MAS Data and Chapter 4, Examples: Two-Color Data for more examples of other plots.

The following set of graphs are examples of the QC Diagnostic plots for the **SurgeryAffyBatch** dataset.

Figure 3.31 displays the image plot for one of the arrays in the study. Each pixel on the image corresponds to a different probe spot on the array. These plots can reveal systematic irregularities due to poor hybridization or problems with the array.

Figure 3.32 displays a plot of the first two principal components for all the arrays. Each point in the plot corresponds to a different array. Different symbols correspond to different experimental conditions.

Figure 3.33 displays an RNA degradation plot. Each line represents the sequence of average expression intensities at each probe location in a probe-set, averaged across all probe-sets on one array. Different colors represent different experimental conditions. Trends in the lines indicate uneven hybridization over the probe-set.

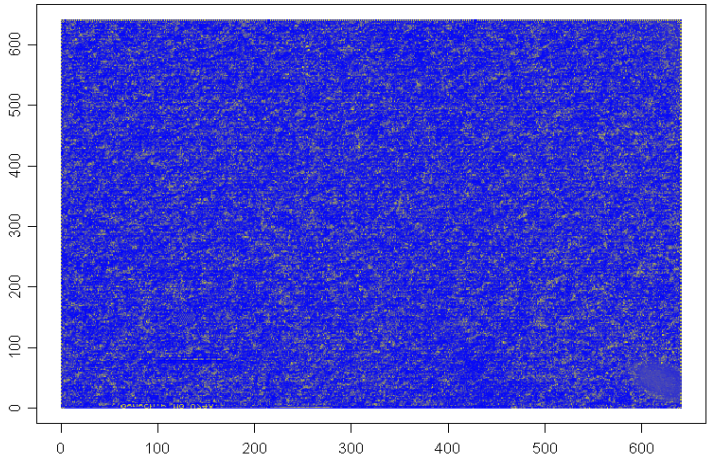


Figure 3.31: *Image plot for one of the arrays in the SurgeryAffyBatch dataset.*

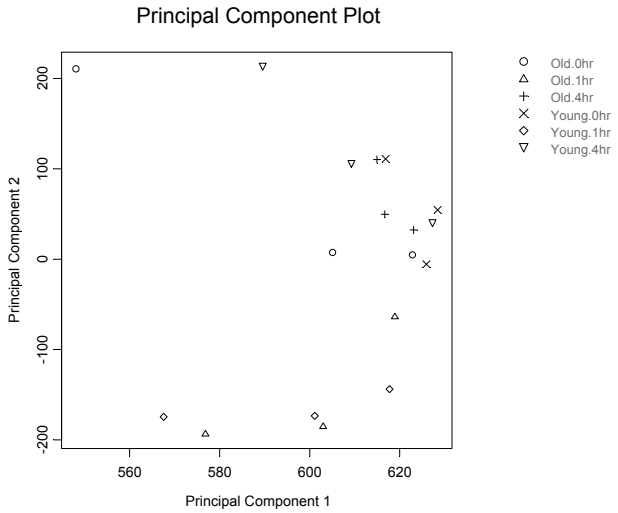


Figure 3.32: *Principal components plot of the experimental conditions in the Surgery study. Different points correspond to different arrays. Plotting symbol indicates experimental condition.*

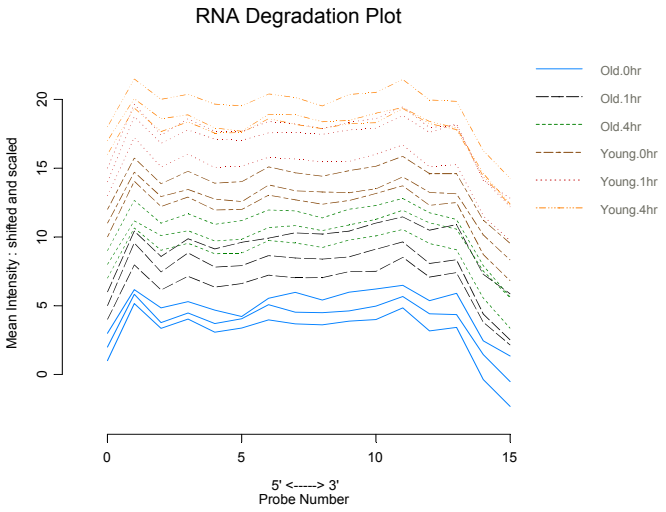


Figure 3.33: *RNA degradation plot*

Normalization

Normalization procedures may be applied to both raw probe-set intensities *and* to summarized expression intensities. For examples of normalizing expression summary data see Chapter 2, Examples: Affymetrix MAS Data and Chapter 6, Pre-Processing and Normalization. In this section we focus on normalizing probe-set data without summarizing it.

Normalization Dialog

Open the **Normalization** dialog by selecting **Normalization** from the **ArrayAnalyzer** drop-down menu.

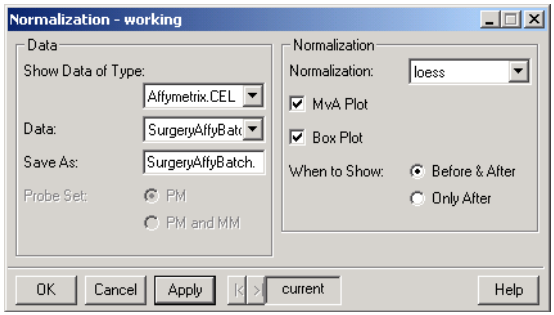


Figure 3.34: *The **Normalization** dialog.*

Select **Affymetrix CEL** from the **Show Data of Type** drop-down list and choose the **SurgeryAffyBatch** data object. Explore the normalization procedures available in the **Normalization** drop-down list. The **quantiles** procedure allows you to normalize only the PM intensities or both PM and MM intensities.

Save As

The **Save As** field takes an object name for saving the normalized **affyBatch** (probe-level) object.

Clicking **OK** creates the normalized **affyBatch** object and plots pre- and post-normalization boxplots for comparison. The plot is on a $\log_2$ scale but the expression intensities are saved on the original (raw) intensity scale. Figure 3.35 displays the results from computing invariantset normalization on probe-level expression intensities.

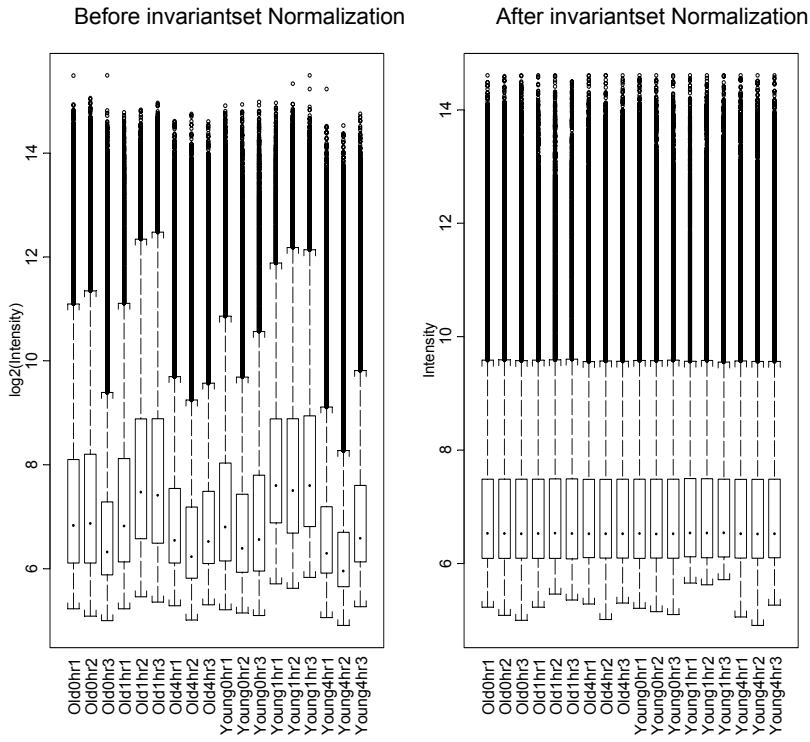


Figure 3.35: *Pre- and post-normalized probe-level data. The normalized intensities are plotted on the log_2 scale.*

Expression Summaries

Once you import the data files you need to convert the raw probe-level expression intensities to expression *summaries* before testing for differential expression. This is usually done in a series of steps including some combination of the following:

- Background correction
- Probe-specific background correction, e.g. subtracting MM
- Normalization
- Summarization of the probe set values into one expression measure. Sometimes standard errors are computed as well.

An assortment of procedures are available for completing these steps. In addition to normalization in the context of summarizing raw expression intensities you can also normalize without the summarization step. For more detail see Chapter 6, Pre-Processing and Normalization.

RMA Summary

In this section we focus on one sequence of steps referred to as robust multichip analysis or RMA for short. This procedure completes the following steps:

1. Probe specific correction of the PM probes using a model based on observed intensity being the sum of signal and (background) noise.
2. Normalization of corrected PM probes using quantile normalization (Bolstad et. al., 2002).
3. Calculation of expression measures using median polish.

This sequence of steps is available by simply selecting the **RMA** checkbox in the upper right corner of the **Affymetrix Expression Summary** dialog. Open the dialog by selecting the **Affymetrix Expression Summary** selection from the **ArrayAnalyzer** menu. Then select the **SurgeryAffyBatch** object in the **CEL Data** drop-down list and select the **RMA** checkbox. The result of the computation is an expression summary object. Save the result as **SurgeryExprSet.rma** by typing it into the **Save As** field.

RMA Output

A sequence of graphs is produced as output by the RMA procedure. Figure 3.37 displays MvA plots created for all the replicates within one treatment condition. The values in the lower left triangle of the matrix are the inter-quartile ranges (IQR) of the values of M across all (summarized) expression values. A small value indicates there is little

difference (on the $\log_2$ scale) for the middle 50% of the expression values for the two chips. For replicate chips there is no *real* differential expression so the IQR is expected to be small.

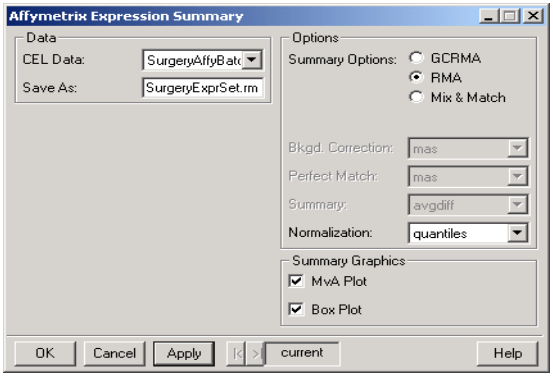


Figure 3.36: *Specifying Robust Multichip Analysis with a single checkbox.*

Figure 3.38 displays boxplots of logged expression summaries for each array. Visual inspection shows the distributions are well aligned at their centers and quartiles. Although normalization may be repeated sequentially to summarized expression intensities there is little need to apply more normalization to `SurgeryExprSet.rma`.

Note

The values displayed in the MvA plots in Figure 3.37 depend on the values used for `.Random.seed`. The samples change from plot to plot so you may see slightly different plots as a result.

Log_2 of Expression Intensities

After applying normalization and summarization procedures to the raw expression intensities a log base 2 transformation is applied. Consequently, the returned summarized object contains expression intensities on a $\log_2$ scale. The log transformation is computed as

$\log_2(\text{E})$ if $\text{E} > 1$

0 if E less than or equal to 1.

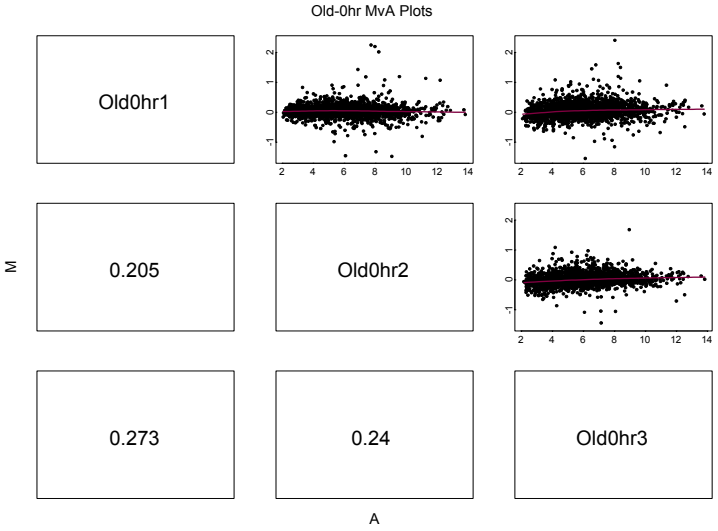


Figure 3.37: *MvA plot for the three replicate samples of old mice measured at 0 hours. The values in the lower left triangle of the plot are the interquartile ranges of M.*

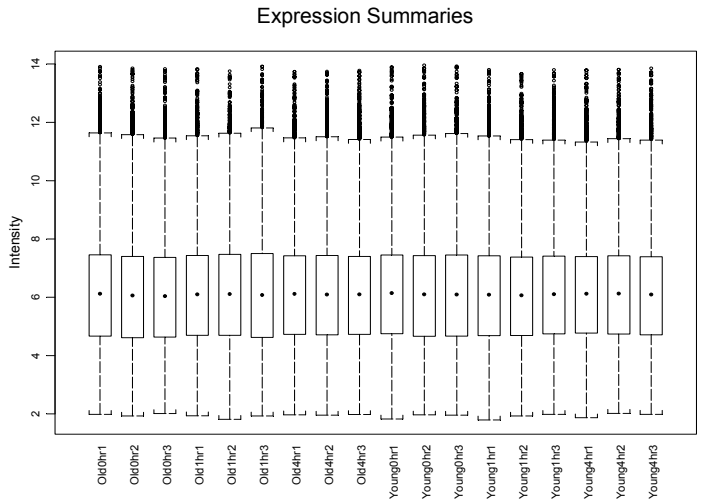


Figure 3.38: *Boxplot of log_2 expression intensities for the 18 samples after applying the composite RMA procedure.*

Differential Expression Analysis

ANOVA Dialog

We are now ready to do differential expression analysis for the two-way design. From the main menu, open the **ANOVA** dialog by clicking **ArrayAnalyzer ▶ Differential Expression Analysis ▶ ANOVA**. The procedures implemented in this dialog provide traditional ANOVA methods with a host of correction procedures to control the Type I Error rate. For more details on the **ANOVA** dialog and Type I Error correction procedures see Chapter 7, Differential Expression Testing

To set up the **ANOVA** dialog, follow these steps:

1. In the **Show Data of Type** field, select **Affymetrix**.
2. In the **Data** field, select **SurgeryExprSet.rma**.
3. The **Array Name** field should be automatically updated to **mgu74av2**.
4. In the **Contrasts** group confirm that the **Baseline** checkbox is checked.
5. Select **Age** as the factor for doing baseline comparisons.
6. From the **Baseline Level** drop-down list select **Young**.
7. In the **Options** group select the BH (Benjamini-Hochberg) FDR adjustment procedure.
8. In the **SaveAs** field in the lower right-hand corner of the dialog **SurgeryANOVABsYoungBH**.
9. Click **OK** to compute the ANOVA results.

.

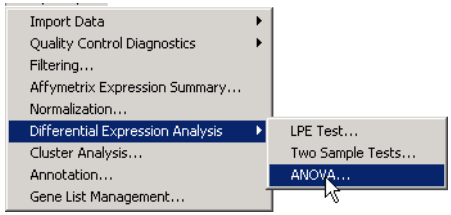


Figure 3.39: *Opening the* ***(Differential Expression Analysis) ANOVA*** *dialog.*

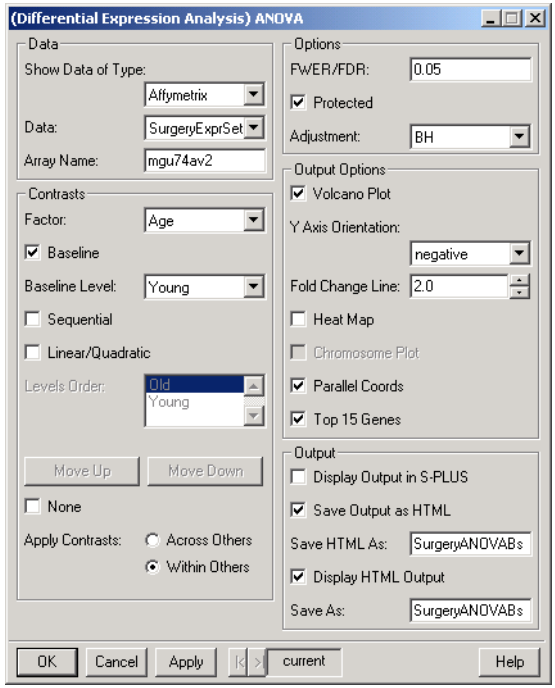


Figure 3.40: *The* ***(Differential Expression Analysis) ANOVA*** *dialog.*

The **Options** group allows you to set the family-wise error rate (FWER) and false discovery rate (FDR) to control the overall Type I error (false positive) rate based on adjusting individual test p-values to account for multiple tests. In our surgery study there are 12,488 genes so the Type I error is substantial without adjusting the p-values.

There are many options for adjusting the p-values to achieve the FWER or FDR you want. Here we set the adjustment procedure to Benjamini-Hochberg (BH) which has the nice property of limiting the Type I error (i.e., false positive rate) to a small percentage of the significant genes.

There are four options in the **Output Options** group:

1. Volcano plot
2. Heat map
3. Parallel Coords
4. Top 15 Genes

Chromosome Plot Not Available for All Chips

Note that chromosome plots are not available for arrays other than **hgu95a**.

Graphical Output

Two examples are shown - the Volcano plot and the Parallel coords plots. Figure 3.41 displays the Volcano plot for the Young versus Old contrast at 1 hour. There are 46 significant genes in the plot resulting from the BH correction. Even with an 8-fold increase in significant genes, the BH correction maintains a low false positive rate of 5%

amongst the significant genes. This translates to, on average, about 2-3 genes not really differentially expressed amongst those genes tagged as significant by the correction procedure.

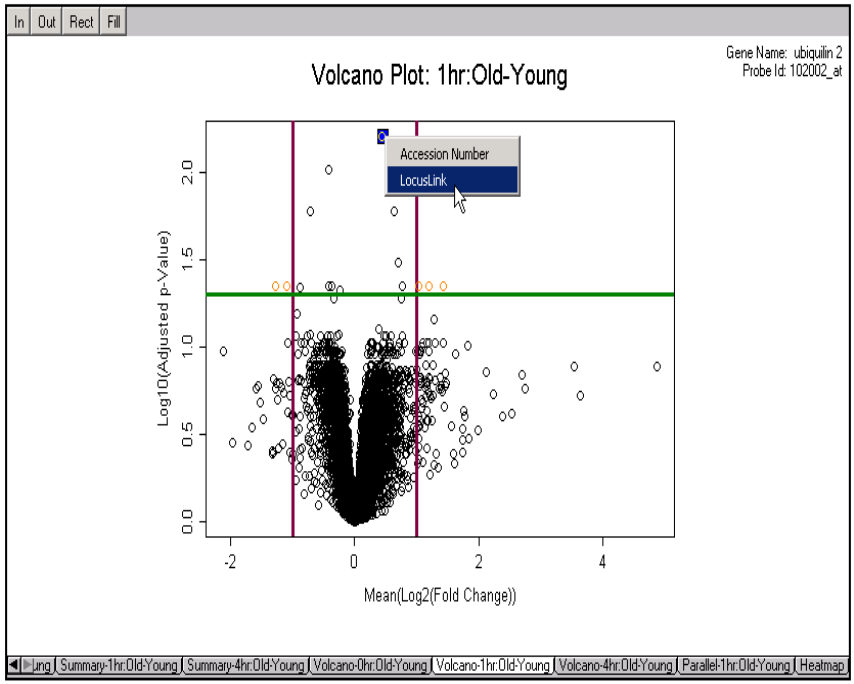


Figure 3.41: *A volcano plot, which is the logarithm of p-value versus fold change. The Benjamini-Hochberg FDR correction method was used for the surgery data at an FDR of 0.05. The BH correction is less conservative than the Bonferroni procedure, yet maintains a small proportion of false positives amongst those genes tagged as significant.*

Figure 3.42 displays the parallel coords plots which show how the expression intensity varies across the treatment conditions for the significant genes.

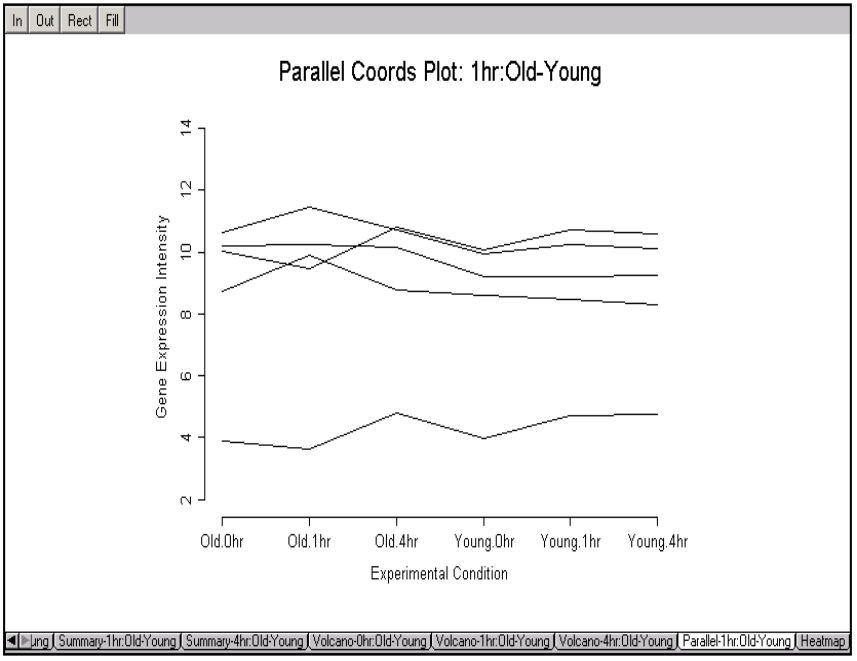


Figure 3.42: *Parallel coords plots of expression intensities for the significant genes. Each line shows the expression intensity profile for a gene across the experimental conditions.*

LPE Test Dialog

An alternative way to think about the analysis is a simple comparison between Young and Old, ignoring the Time component. This then becomes a two-sample problem. S+ARRAYANALYZER has two dialogs for two-sample problems: The **LPE Test** dialog and the **Two Sample Tests** dialog. See Chapter 7, Differential Expression Testing for more details.

To apply the LPE test procedure to this experimental open the dialog by clicking **ArrayAnalyzer ▶ Differential Expression Analysis ▶ LPE Test**. Select Affymetrix data and the **SurgeryExprSet.rma** data set. Now choose the **Age** factor creating comparisons. Running the LPE Test creates a volcano plot, a heat map plot and variance plots as well as a summary table of most significant genes.

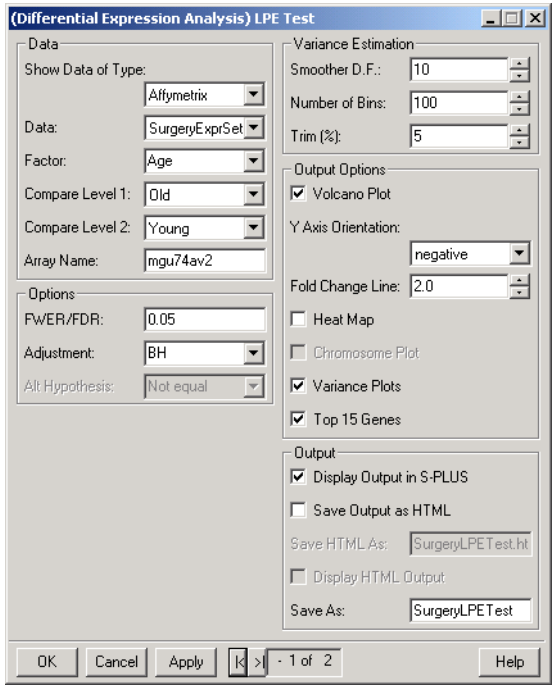


Figure 3.43: *The* `SurgeryExprSet.rma` *object used in the* ***LPE Test*** *dialog, used in the two-sample problem using the mouse surgery example data.*

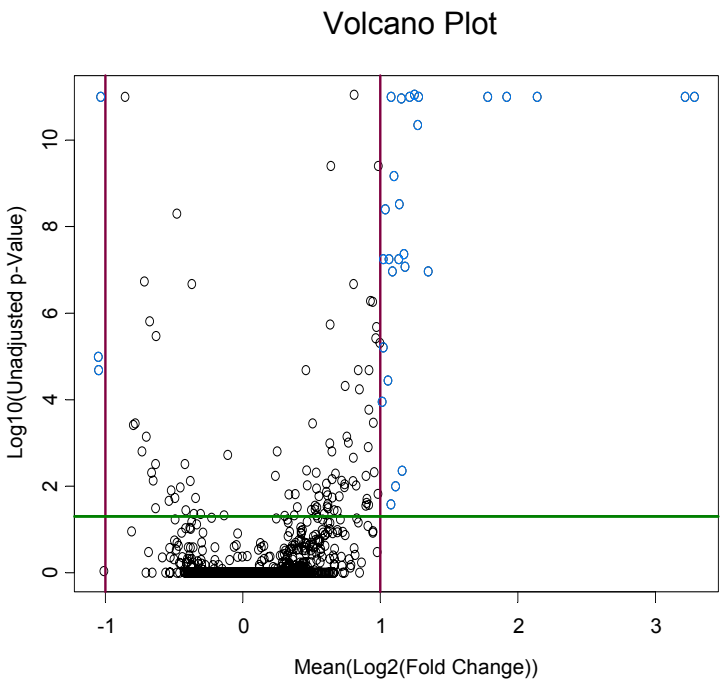


Figure 3.44: *LPE test comparing Young and Old ignoring Time for the SurgerExprSet.rma data set.*

Two Sample Test Dialog

Similar results may be obtained from the **Two Sample Tests** dialog but for different analysis methods.

Graphical Annotation

Clicking one of the hyper linked points in either the volcano plot or the heat map (when you are view the HTML graphical output) pops up a menu for selecting the database to query for annotation information. Selecting either one opens an HTML page in your default web browser displaying a brief description of the gene with a hyper link to more detailed information. Figure 3.45 shows an example page from LocusLink with annotation for a differentially expressed gene.

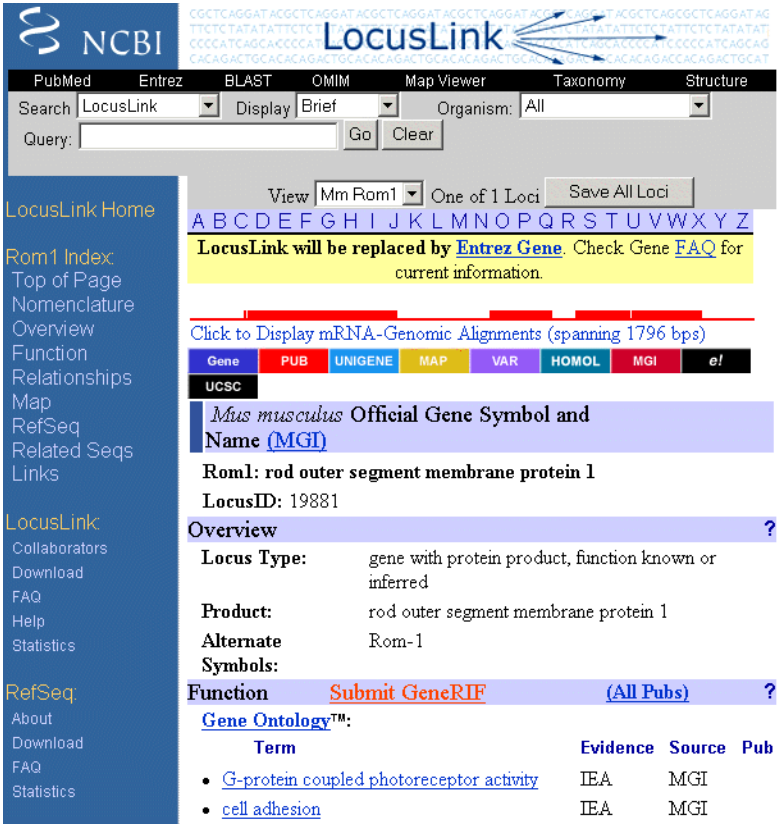


Figure 3.45: *Locus Link annotation page resulting from graphical annotation query.*

Gene List Management

The **Gene List Management** dialog allows you to merge gene lists from testing different contrasts in an analysis *or* from different analyses. The Surgery experiment is a two-way design with several contrasts of interest listed in Table 3.3.

The results of the ANOVA are contrasts of Old and Young for each time point. Use the Gene List Management dialog to combine the gene lists from each contrast.

Open the **Gene List Management** dialog by clicking **ArrayAnalyzer ► Gene List Management**. In **Data Group 1** select **DiffExpr Test** from the **Show Data of Type** drop-down list. Then select the **SurgeryANOVABsYoungBH** object from the **Data** drop-down list and select one of the contrasts listed in Figure 3.46. Start with **1hr:Old-Young**. In **Data Group 2** also select **DiffExprTest** as the data type and choose **4hr:Old-Young** as the contrast. Now, check the **Union** radio button in the **Output Options** group, confirm that the **Venn Diagram** checkbox is checked and type in **SurgeryGeneList** in the **Save As** field to save the resulting gene list.

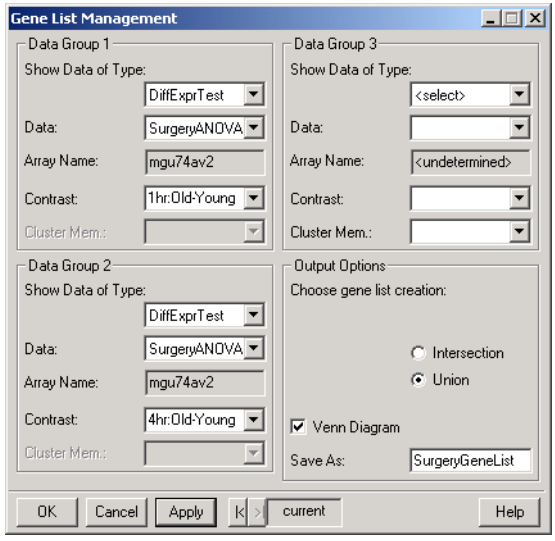


Figure 3.46: *The **Gene List Management** dialog.*

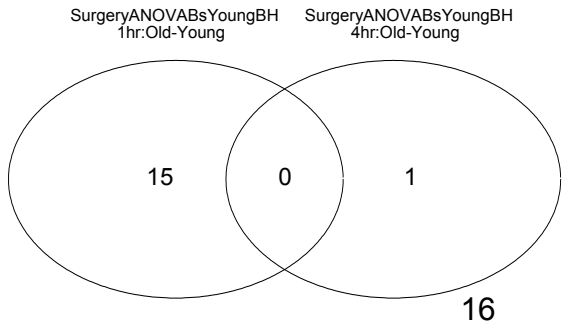


Figure 3.47: *Venn diagram comparing gene lists for two different contrasts of the Surgery data set.*

Annotation

In addition to the annotation described in section Graphical Annotation on page 117, there is a dialog with more general annotation capabilities. To open the **Annotation** dialog, click **ArrayAnalyzer ► Annotation**. The resulting dialog has two tabs:

1. **General Options**.
2. **Filtering Options**.

There are many options available on the **Annotation** dialog, but, for this first example, we'll focus on generating the annotation for the significant genes in **SurgeryANOVABsYoungBH**. For a more thorough description of the **Annotation** dialog see Chapter 9, Annotation and Gene List Management.

To annotate only the significant genes identified in **SurgeryANOVABsYoungBH**, select **DiffExprTest** from the **Show Data of Type** list and select **SurgeryANOVABsYoungBH** from the

Data list. Note the public annotation databases available from the GUI in the **General Annotation** group. For the example, we leave the three defaulted databases checked. Figure 3.48 displays the resulting settings.

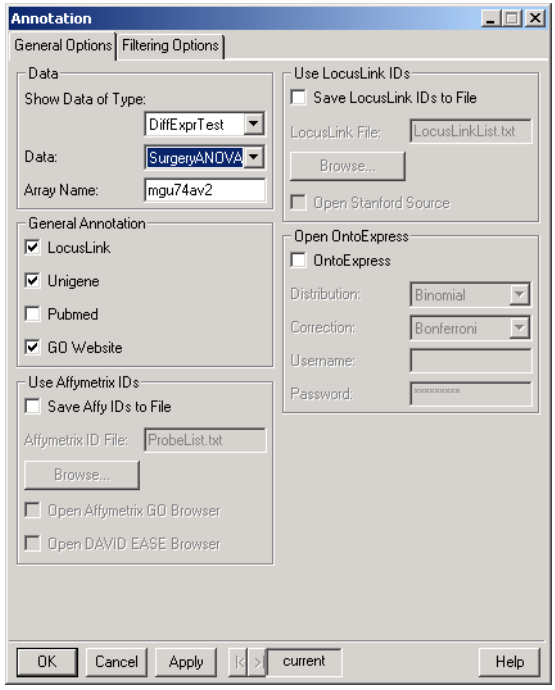


Figure 3.48: ***General Options*** *settings for annotation of the genes identified by the* ***SurgeryANOVABsYoungBH*** *ANOVA.*

Now on the **Filtering Options** tab in the **Contrast Filtering** group select the **1hr:Old-Young** contrast and check the **Significant genes** checkbox. Clicking the **Recalculate** button in the **Gene Sort Order Options** group will show you how many genes are selected by the filtering. Note the **Limit number of genes to** field which puts a cap on the number of gene ID's that will be sent to the databases for annotation extraction. Figure 3.49 displays the resulting settings.

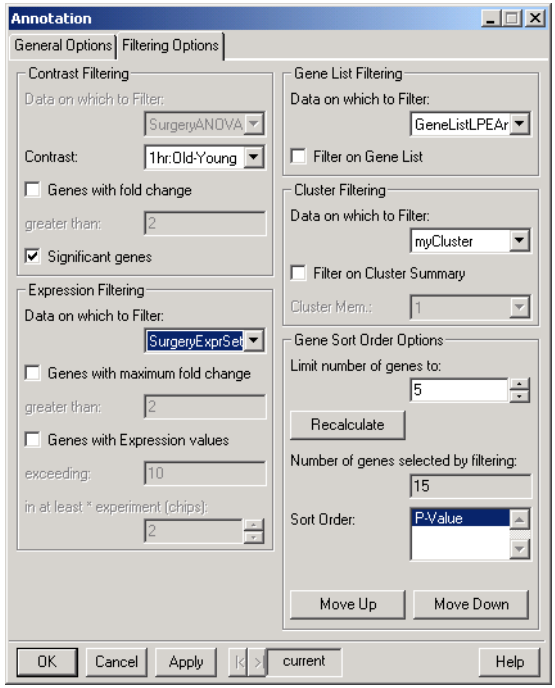


Figure 3.49: ***Filtering Options*** *settings for annotation of the genes identified by the* ***SurgeryANOVABsYoungBH*** *ANOVA.*

Clicking **OK** now generates the annotation lists from the databases selected. Figure 3.50 displays a summary table for the top significant genes identified by the ANOVA tests for the **1hr:Old-Young** contrast.

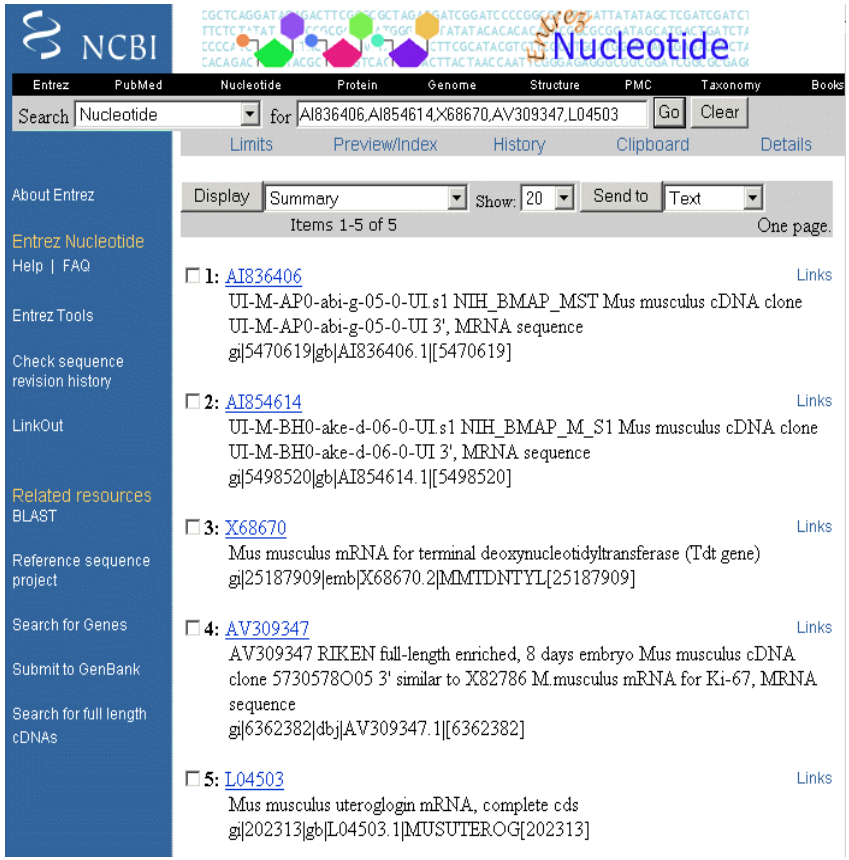

NCBI
Nucleotide
Entrez PubMed Nucleotide Protein Genome Structure PMC Taxonomy Books
Search Nucleotide for AI836406,AI854614,X68670,AV309347,L04503 Go Clear
Limits Preview/Index History Clipboard Details
About Entrez
Entrez Nucleotide
Help | FAQ
Entrez Tools
Check sequence revision history
LinkOut
Related resources
BLAST
Reference sequence project
Search for Genes
Submit to GenBank
Search for full length cDNAs
Display Summary Show: 20 Send to Text
Items 1-5 of 5
One page.
1: AI836406 Links
UI-M-AP0-abi-g-05-0-UI.s1 NIH_BMAP_MST Mus musculus cDNA clone UI-M-AP0-abi-g-05-0-UI 3', MRNA sequence
gi|5470619|gb|AI836406.1|[5470619]
2: AI854614 Links
UI-M-BH0-ake-d-06-0-UI.s1 NIH_BMAP_M_S1 Mus musculus cDNA clone UI-M-BH0-ake-d-06-0-UI 3', MRNA sequence
gi|5498520|gb|AI854614.1|[5498520]
3: X68670 Links
Mus musculus mRNA for terminal deoxynucleotidyltransferase (Tdt gene)
gi|25187909|emb|X68670.2|MMTDNTYL[25187909]
4: AV309347 Links
AV309347 RIKEN full-length enriched, 8 days embryo Mus musculus cDNA clone 5730578O05 3' similar to X82786 M.musculus mRNA for Ki-67, MRNA sequence
gi|6362382|dbj|AV309347.1|[6362382]
5: L04503 Links
Mus musculus uteroglogin mRNA, complete cds
gi|202313|gb|L04503.1|MUSUTEROG[202313]

Figure 3.50: *Annotation summary from Entrez Gene.*

REFERENCES

Bolstad, B. M. Irizarry, R. A., Astrand M. and Speed T. P. (2002). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* 19(2): 185-193.

Irizarry, R. A., Bolstad B. M., Collin, F., Cope, L. M., Hobbs, B., and, Speed, T. P. (2002). Summaries of Affymetrix GeneChip Probe Level Data. *Nucleic Acids Research.* Vol. 31, No. 4 e15

Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U., Speed, T. P. (2003). Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data. *Biostatistics 4:249-264.*

EXAMPLES: TWO-COLOR DATA

4

Two-Color Data Analysis Workflow **126**

Two-Sample Design **127**
- Mutant Zebra Fish Data 127
- Importing Data 128
- Normalization 138
- Differential Expression Testing 142

Two-Way Reference Design **147**
- Malaria Parasite Data 147
- Importing Data 148
- Quality Diagnostics 151
- Filtering 155
- Quality Diagnostics Revisited 157
- Normalization 158
- Clustering Expression 161
- Differential Expression Analysis 164
- Gene List Management 166
- Annotation 168

From the Command Line **174**
- Importing Data 174
- Filtering Out Controls 180
- Quality Diagnostics 180
- Normalization 183
- Differential Expression Testing 187

TWO-COLOR DATA ANALYSIS WORKFLOW

The process of analyzing differential expression for custom cDNA arrays can be done through the S+ARRAYANALYZER menu and dialogs. To obtain differential expression test results from two-color cDNA microarray data, we go through four fundamental steps:

1. Importing and filtering the data.
2. Background adjustment and normalization.
3. Differential expression analysis.
4. Gene list management and annotation.

In addition S+ARRAYANALYZER provides methods for quality diagnostics and clustering to round out its feature list.

The two examples of cDNA data we use are summarized (means and medians) across all spots with identical probes. When we import the data, we specify background intensity columns so adjustment for background intensity levels can be made prior to normalization and testing.

TWO-SAMPLE DESIGN

S+ARRAYANALYZER has an assortment of procedures for doing two-sample differential expression analysis. They include the t-test with or without assuming equal variance for the groups, the Wilcoxon rank-sum test and several permutations tests. In addition the Local Pooled Error (LPE) test procedure, which produces improved error estimates when there is little replication in the design, is implemented for two-sample problems. All these procedures are suitable for doing simple comparisons between two groups - treatment versus control, tissue 1 versus tissue 2, etc. For more details about two-sample procedures see Chapter 7, Differential Expression Testing.

Mutant Zebra Fish Data

In this section, we examine two-color microarray data from a developmental biology experiment. The data are included with the Bioconductor distribution and were originally provided by Katrin Wuennenberg-Stapleton from the Ngai Lab at UC Berkeley. The experiment was designed to study the early development of vertebrates using zebrafish as a model organism. Zebrafish embryos from two genetic strains were used - a *swirl* mutant and a normal *wild-type*. The goal was to identify genes with differential expression between the two strains. Refer to the `swirl` help file for more details.

The experiment consisted of two sets of dye-swap experiments resulting in a total of four arrays. Each pair of experiments swapped the color labels between the swirl and wild-type samples. Table 4.1 details the experimental conditions and the associated data files.

Table 4.1: *Experimental design and file association for the swirl cDNA experiment.*

Cy3	Cy5	Array	File Name
`swirl`	`wild-type`	1	**swirl.1.spot**
`swirl`	`wild-type`	3	**swirl.3.spot**
`wild-type`	`swirl`	2	**swirl.2.spot**
`wild-type`	`swirl`	4	**swirl.4.spot**

Importing Data

To import two-color cDNA data, go to the main S-PLUS menu and select **ArrayAnalyzer ► Import Data ► From Two Channel**.

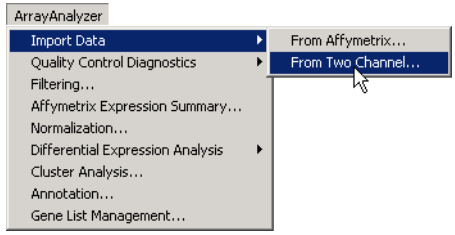


Figure 4.1: *Menu selection to import two-color cDNA data.*

(Import Data) From Two Channel Dialog

This launches the (**Import Data**) **From Two Channel** dialog, with the **File Selection** page displayed. The primary task of the import process is to associate data files with experimental conditions and select the variables and corresponding columns that are used in the data analysis.

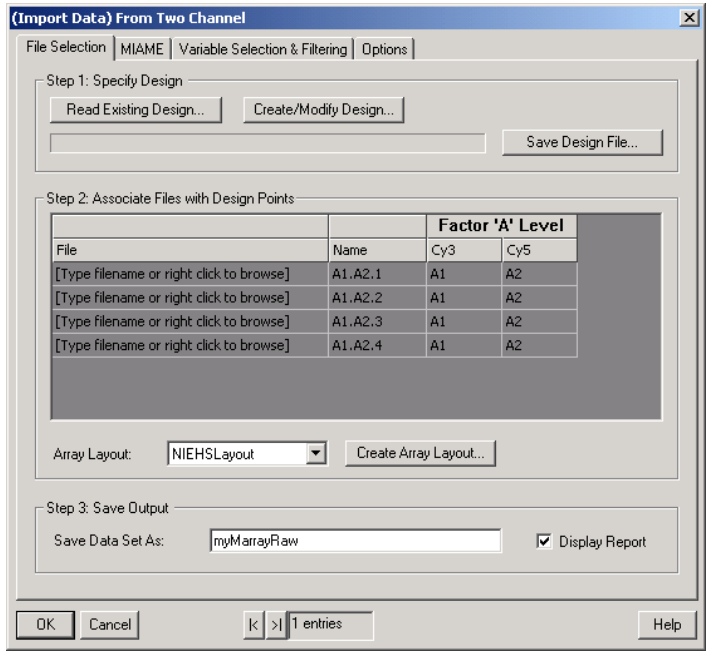


Figure 4.2: *The **Import cDNA Data** dialog.*

The (**Import Data**) **From Two Channel** dialog has four pages:

- **File Selection** This page *must* be completed in order to create a data object for continued analysis.
- **MIAME** Completing this page is optional but highly recommended because information on the **MIAME** page is used for labeling tables and graphs.
- **Variable Selection & Filtering** Red and green foreground colors are required. Background colors are optional.
- **Options** This page provides options for specifying the number of header lines to skip and the delimiter used in the data file.

Data import is accomplished in three steps:

1. Create the experimental design.
2. Associate files with design points.
3. Save data set as an S-PLUS object.

Step 1: Create The Experimental Design

Before you associate data files with experimental conditions, you must to set up the experimental design in S+ARRAYANALYZER. Start by opening the **Create/Modify Design** dialog - **Step 1** on the (**Import Data**) **From Two Channel** dialog.

For this simple two-sample experiment most of the defaults work as they are. The **Number of Arrays** and the **Number of Factors** have defaults of four and one, respectively, which correspond to this experiment.The settings to change are:

1. Set the **Factor Name** to **Zebra Fish**
2. Set the **Level Values** to **Swirl** and **WildType**.

Level Values Format
The **Level Values** strings you enter can contain no spaces. In this example, we use **WildType** not **Wild Type**.

3. In the **Design Type** group, ensure that the **Two Sample** radio button is selected and the **Dye Swap** checkbox is checked.

4. Clicking **OK** on the **Create/Modify Design** dialog generates the experimental design points that populate the file selection grid. You are now ready for **Step 2: Associate Files with Design Points**.

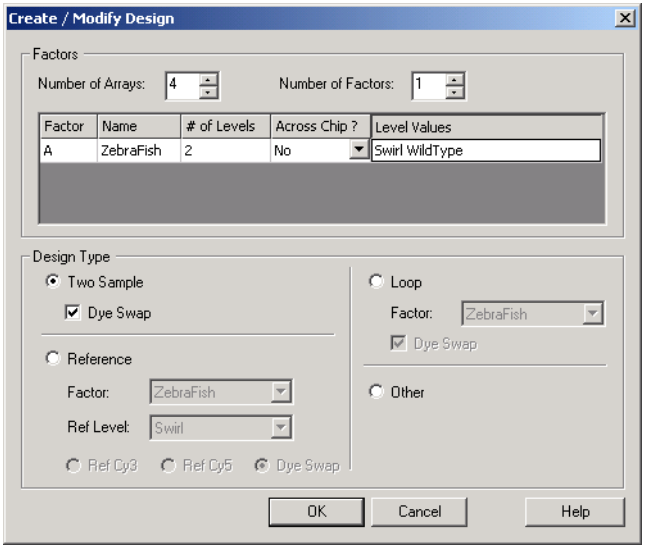


Figure 4.3: *The **Create/Modify Design** dialog set up for the Zebrafish study.*

Step 2: Associate Files with Design Points

Once the design is complete, click **OK** to copy it into the **File Selection** tab of the (**Import Data**) **From Two Channel** dialog. Notice that the number of rows for the **File Selection** box matches the number of arrays specified on the **Create/Modify Design** dialog. Furthermore, values for the factor levels have been written into the **Factor** columns to facilitate associating files with design points. If the experiment is balanced the factor level settings will be exactly as needed. However, the level values can be reset when the experiment is unbalanced or if you prefer an order different from the default.

To associate data files with the design points, in the uppermost **File** field, type in the path to the file or right-click and select **Browse for file**. Repeat this step until you've selected all your files being careful to match them with the design points.

You can find the swirl example data by navigating to your **splus62/module/ArrayAnalyzer/examples** directory, selecting **SPOT** (***.spot**) as the **Files of type**, and selecting **swirl.1.spot**. Repeat for the other three **.spot** files, entering one file per cell, as shown in Figure 4.4.

Selecting Multiple Files
You can select multiple files when you're in the browser. The order you select them will be the order they fill in the file selection grid.

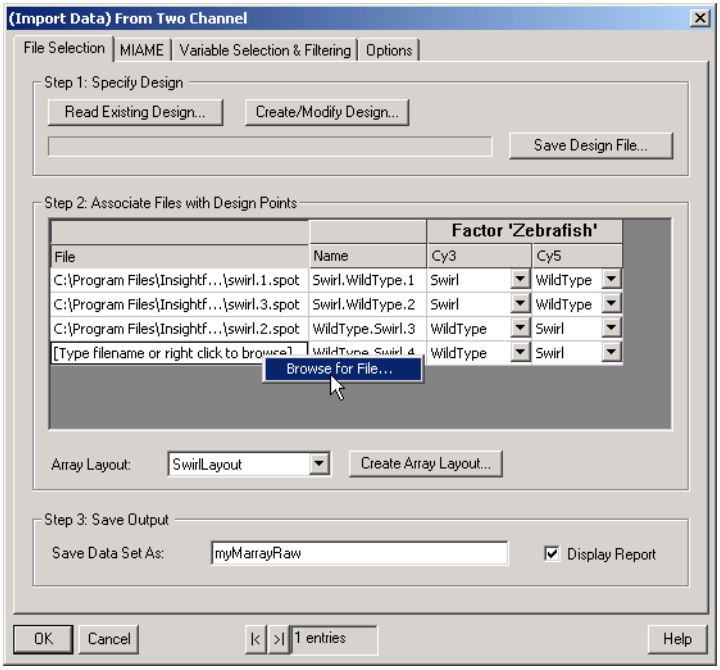


Figure 4.4: *Selecting files for import.*

Creating the Layout

The second part of the file association task is specifying a layout file. To specify the layout for a cDNA array, select an S-PLUS layout object that has been previously created or create a new one.

Create a layout object for the swirl example by clicking the **Create Array Layout** button just beneath the file selection grid.

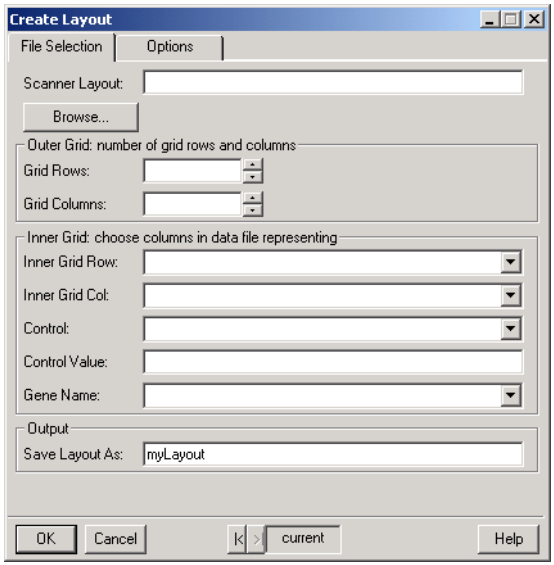


Figure 4.5: *The **Create Layout** dialog.*

The **Create Layout** dialog requires you to fill in the following information:

1. **Scanner Layout:** A layout file name. Enter the path in the **Scanner Layout** field or click the **Browse** button to locate the layout file. The file should be a text file with columns for the gene names and control indicator.

   For the example, navigate to the **fish.gal** file located in your **splus62\modules\ArrayAnalyzer\examples** directory.

2. **Outer Grid:** The print-tip group grid size in rows and columns. The schematic in Figure 4.6 represents a cDNA array. The spots are arranged in large *blocks* or print-tip groups. Within each print-tip group are *spots* where the cDNA is fixed. To specify the array layout, you must specify the size and arrangement of both the print-tip groups (the outer grid)

and the spot matrix (the inner grid) within each group. It is assumed that the spot matrix size is the same for each print-tip group.

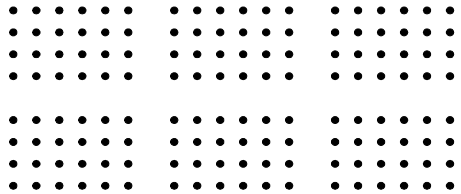


Figure 4.6: *Schematic of a cDNA array layout*

The *outer grid* refers to the layout of the print-tip groups. You specify this layout in terms of rows and columns. In the schematic in Figure 4.6, there are *two* rows and *three* columns of print-tip groups. In the swirl example, there are four rows and four columns.

Enter **4** for both the **Grid Rows** and **Grid Columns** in the **Outer Grid** group.

Single Large Print-Tip Group
Note that some arrays have a single large print-tip group covering the entire array. In this special case, the outer grid has just one row and one column.

3. **Inner Grid**: The inner grid is specified by selecting columns that correspond to row and column identifiers in the layout file. The appropriate column names may be detected automatically and filled-in for you. If not select the appropriate column name for each of the **Inner Grid Row** and **Inner Grid Column** fields. For our example they are automatically detected and filled in as **Row** and **Column**, respectively.
4. **Control**: A column indicating which spots are control spots. Enter **ID** for the Zebrafish example.
5. **Control Value**: Values which label the control spots. In some cases this may be the single value “control”. In others, it may be the numeric values -1 and 1 indicating positive and negative controls.

6. **Gene Name**: A column with unique gene or probe names. Enter **Name** for the Zebrafish example.

7. **Save Array Layout As**: A name for the saved layout object. You need to name the layout object which can be reused for all arrays with the same layout.

   Enter **SwirlLayout** as the object name in the **Save Array Layout As** field, which is used when control returns to the (**Import Data**) **From Two Channel** dialog.

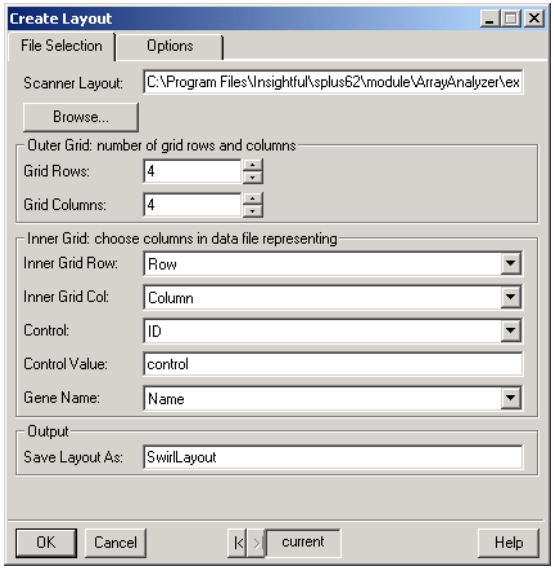


Figure 4.7: *The completed layout for the swirl example.*

When you finish entering layout information in the dialog, your input should look like what is shown in Figure 4.7.

Click **OK** to create and save the layout object. This returns you to the **Import cDNA Data** dialog (Figure 4.4), where you can finish the import dialog.

Step 3: Save Data Set As

To save the data object, type a name in the **Save Data Set As** field near the bottom of the dialog (**Step 3: Save Output**). Remember this name. It is used in other analysis steps such as quality checks, filtering and normalization. For our example, enter **SwirlMarrayRaw** as the

object name. The **Display Report** checkbox indicates whether or not to print summary information, resulting from reading the data, into an S-PLUS report window.

Saving the Design Once you've entered all the information on this tab you can save it for later use by clicking the **Save Design File** button at the top of the dialog. A **.txt** file is written to the directory of your choice with number of factors, number of levels, repetitions and the full path file names and their associated design points.

Reading Designs You can reuse the *design* file for another experiment with the same design by modifying the file locations and names and factor levels as needed. In fact, if you have many arrays in your experiment you can create a file with all the design content and read it with the **Read Existing Design** button which will set the number of arrays and fill the file name fields along with their associated design points.

MIAME Page MIAME is an acronym for Minimal Information About a Microarray Experiment. This information can be entered on the second page of the (**Import Data**) **From Two Channel** dialog. This information is not required, but it is stored on the resulting object to identify the source of the data. Once you've entered MIAME information for any experiment, the first three fields are saved and are filled automatically the next time you open this dialog. The MIAME tab of the (**Import Data**) **From Two Channel** dialog is displayed in Figure 4.8.

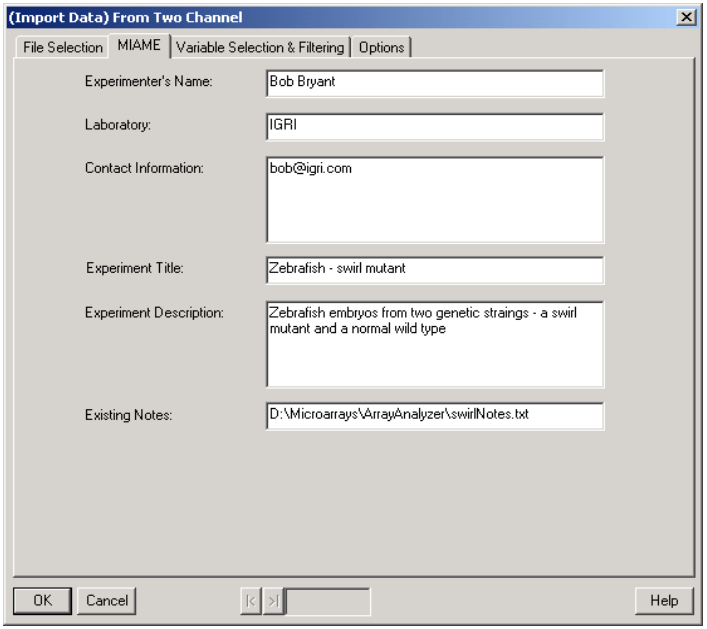


Figure 4.8: *Entering chip information in the **MIAME** page.*

Variable Selection & Filtering Page

The third page in the **Import cDNA Data** dialog is for variable and row selection. There are two required fields on this page, **Green Foreground** and **Red Foreground**, where you must select the columns containing green and red foreground intensities. Select the columns from the drop-down list of variable names, which is populated using the column headers in the imported data files.

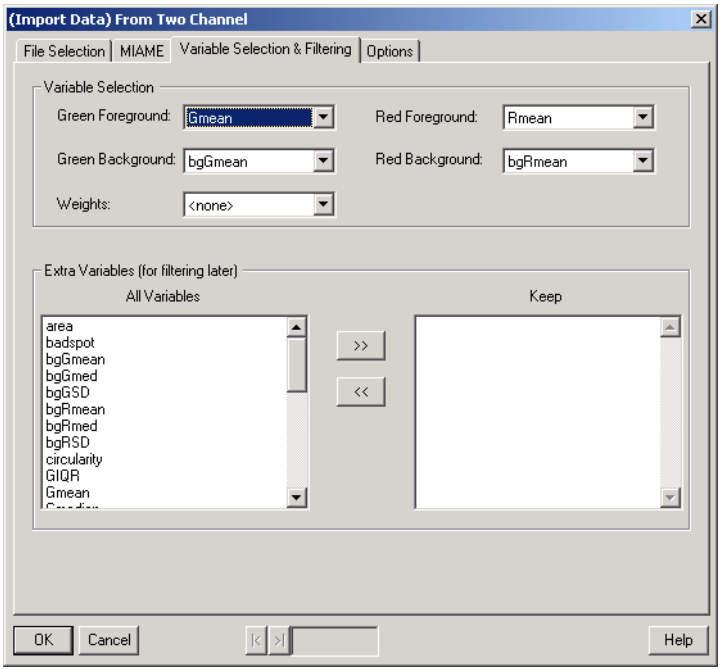


Figure 4.9: *The **Variable Selection & Filtering** page allows you to set the variable and row selections.*

In this example, select **Gmean** as the **Green Foreground** and **Rmean** as the **Red Foreground**, to complete the required fields. Optionally select **bgGmean** as the **Green Background** and **bgRmean** for the **Red Background**.

The **Weights** field is for specifying a column of spot *quality* weights. These weights are used in subsequent computations to down-weight poor quality spots during normalization. See Chapter 6, Pre-Processing and Normalization for more detail.

Clicking OK

Once you complete the **Variable Selection & Filtering** page, click **OK** to begin importing the files. The object resulting from the import step is of class `marrayRaw`, which is saved as an S-PLUS object with the name you entered on the first page of the (**Import Data**) **From Two Channel** dialog, **SwirlMarrayRaw**.

Other Data Formats

Some scanning equipment generates layout information as part of the data file. For example, some scanners generate expression intensity files with columns containing "Row" and "Column" layout information for the spots on the array. When your data files are organized this way, you should be able to read the data through the GUI by selecting one of the (data) files as the layout file for the **Create Layout** dialog and then *reusing* that same file on the **File Selection** page of the (**Import Data**) **From Two Channel** dialog. The **Create Layout** dialog will only pick up the layout, probe names and control information. When you use the file the second time on the **File Selection** page the import operation will pick up the expression intensity columns.

Single Grid Arrayers

Some arrayers don't have a double grid layout as we describe in the swirl example. Agilent with its Inkjet technology for printing arrays produces *one* large spot matrix. In this case set the **Grid Rows** and **Grid Columns** to one on the **Create Layout** dialog.

Normalization

Normalization is designed to remove artifacts and systematic variation resulting from the preparation and measurement process. The goal is to remove variability not due to differential expression so that differential expression is estimated accurately for each gene. Note that we need to be careful not to normalize so aggressively as to wash out signal. For cDNA data, normalization corrects for various types of dye bias as well as print-tip and substratum irregularities. Some examples include the following:

1. Different labeling efficiencies and scanning properties of the Cy3 and Cy5 dyes.
2. Print-tip effects.
3. Spatial (within-slide) effects.
4. Between slide effects.

Furthermore, normalization allows for the use of control spots on the array or spiked into the mRNA samples. In Chapter 6, Pre-Processing and Normalization we provide more detail on different methods of normalization. Here we list them briefly and work through a simple example.

The Normalization Dialog

To normalize cDNA data, go to the main S-PLUS menu and select **ArrayAnalyzer ► Normalization**.

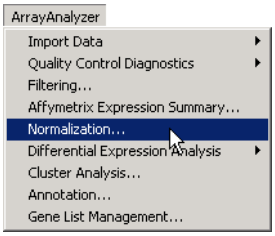


Figure 4.10: *Selecting **Normalization** from the main S-PLUS menu.*

Show data of type

Select the type of data you are normalizing. In this example, select **Two Channel** in the **Show Data of Type** field for the swirl example, as shown in Figure 4.11.

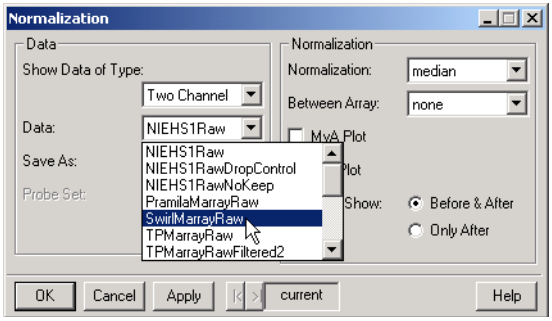


Figure 4.11: *Selecting **cDNA** before selecting for a list of cDNA data.*

Data

In the **Data** field, select **SwirlMarrayRaw** from the drop-down list as the object of class `marrayRaw` created during the import step.

Save As

In the **Save As** field, the name **SwirlMarrayRaw.norm** will be generated by default. You can edit the name in this field if you wish. Our example uses the default object name for the normalized expression data.

Normalization

Now set the other options on the right side of the **Normalization** dialog. The normalization methods are listed in the **Normalization** drop-down list:

- median
- loess
- twoD
- printTipLoess
- scalePrintTipMAD
- global MAD
- printTipMAD

In this example, select **scalePrintTipMAD** as the normalization method. Select the **Box Plot** check box, the **MvA** check box and the **Before & After** radio button for pre- and post-normalization plots.

Click **OK** or **Apply** to produce the normalized data and create the pre- and post-normalization plots, shown in Figures 4.12 and 4.13.

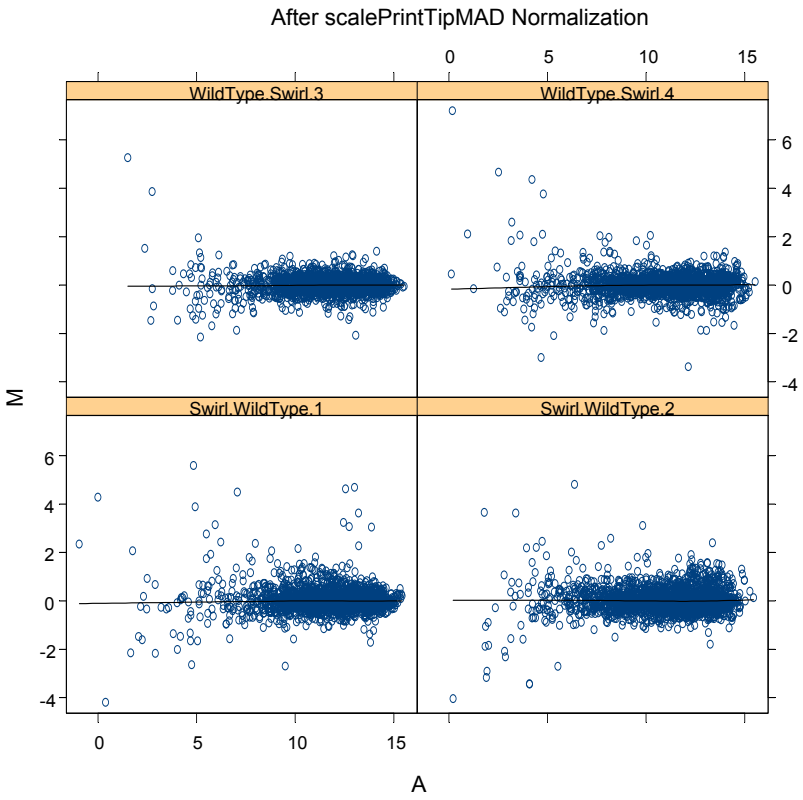


Figure 4.12: *MvA plot of normalized data.*

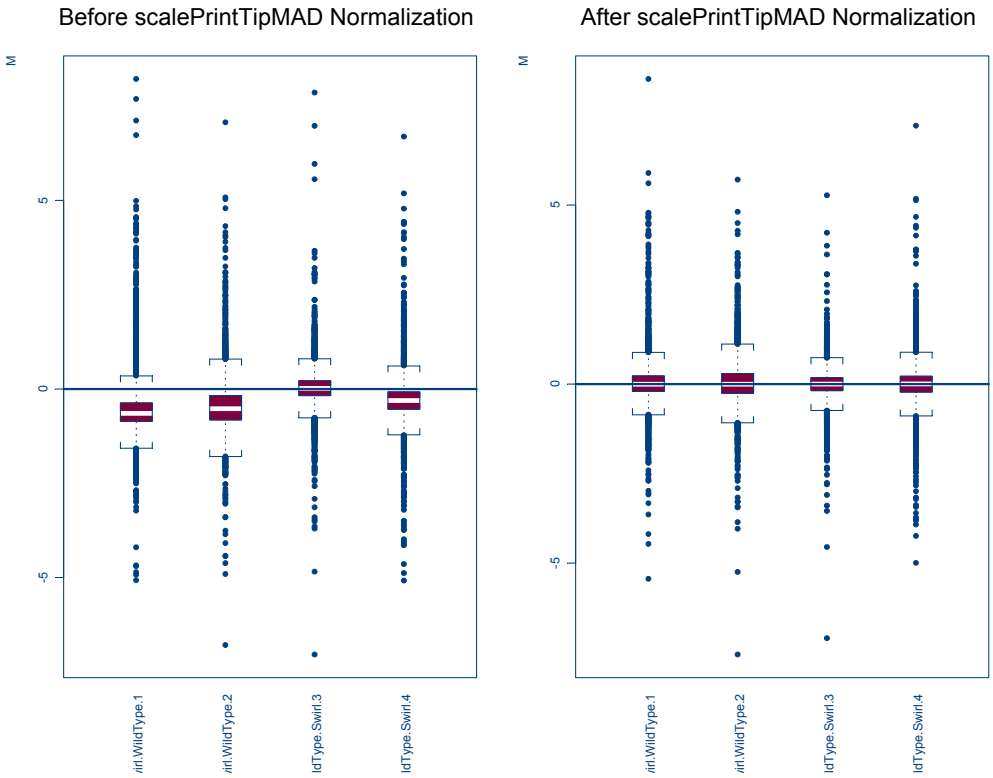


Figure 4.13: *Boxplots of before and after scalePrintTipMAD-normalized swirl data*

Differential Expression Testing

Open the (**Differential Expression Analysis**) **Two Sample Tests** dialog by clicking **ArrayAnalyzer ► Differential Expression Analysis ► Two Sample Tests**. The dialog provides traditional testing methods (e.g., paired-t test, Wilcoxon test) with a host of correction methods to control the family-wise error rate and false discovery rate (Type I Error).

To set up the **Two Sample Tests** dialog, follow these steps:

1. In the **Show Data of Type** field, select **Two Channel**.
2. In the **Data** field, select **SwirlMarrayRaw.norm**
3. For **Compare Level 1** and **Compare Level 2** set the values to **Swirl** and **WildType**, respectively.

4. The **Array Name** field should be automatically updated to **<undetermined>**.
5. Deselect the **Display Output in S-PLUS** and select **Save Output as HTML** and **Display HTML Output** checkboxes.
6. To save the test results enter **SwirlMultTest** in the **Save As** field of the **Output** group.

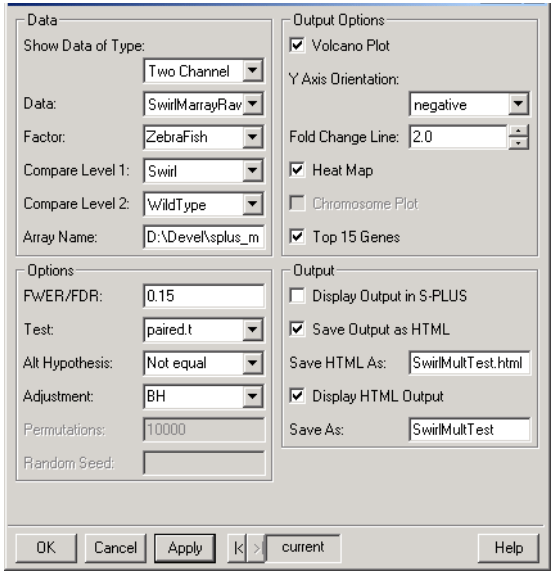


Figure 4.14: *The **Two Sample Test** dialog for the `SwirlMarrayRaw` expression object.*

The Options Group

The **Options** group allows you to specify, the statistical test and alternative hypothesis, a procedure for controlling the family-wise error rate (FWER) or the false discovery rate (FDR) and the maximum error rate. Additionally, certain procedures estimate statistical tests and p-values by permutation sampling. For these procedures you can also set the maximum number of permutations and a random seed for reproducibility and testing.

FWER

FWER and FDR The **Options** group allows you to set the family-wise error rate (FWER) or the false discovery rate (FDR) to maintain an overall Type I error rate (false positive rate) based on adjusting individual test p-values to account for multiple tests. In our swirl/mutant example there are 8448 genes so the increase in Type I error is substantial without adjusting the p-values.

Tests

You can chose the type of test you want to perform: paired-t (the default for two color arrays), Welch's t for unequal variance, an equal variance t and the nonparametric Wilcoxon test. For the swirl example, leave the setting as the default paired-t test.

Alternative Hypothesis

The **Alternative Hypothesis** drop-down list provides three options:

1. Greater than
2. Less than
3. Not equal (default)

These hypotheses refer to the alternative to the Null Hypothesis for the statistical tests that there is no differential expression. Significant differential expression for any given gene means that Compare Level 1 (Swirl) is greater than, less than or not equal to Compare Level 2 (WildType).

Leave the default setting of **"Not equal"** for the swirl example.

Adjustment

There are many options for adjusting the p-values to achieve the FWER. We describe them in more detail in the Differential Expression Testing chapter. Typically we start with the default Bonferroni procedure but instead we select the BH (Benjamini-Hochberg) procedure which is less conservative than Bonferroni and controls the false discovery rate rather than the family wise error rate. See Chapter 7, Differential Expression Testing for more detail.There is little expression activity in this study so set the **FWER/FDR** to 0.15 to be less conservative.

There are three options in the **Output Options** group to display any of the following: a volcano plot, a heat map, or the top-15 gene list.)

Clicking **OK** or **Apply** produces the output plots, which are discussed in the following section.

Volcano Plot

A *volcano* plot displays the logarithm of p-value versus fold change, as shown in Figure 4.15. The vertical lines indicate fold change values of plus or minus two, and the horizontal line indicates a significant LPE Test p-value after doing the Bonferroni correction. Points located in the lower, outer sextants are those with large absolute fold change and small (significant) p-value. Each of those points is active so you can click an individual point to access annotation information from LocusLink or GenBank.

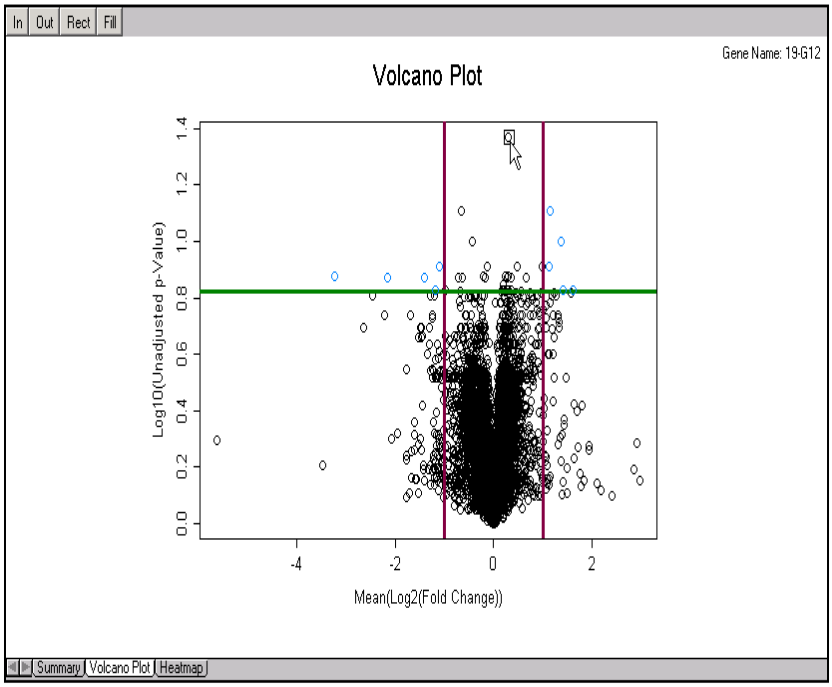


Figure 4.15: *A volcano plot, which is the logarithm of the adjusted p-value versus fold change.*

Heat Map

A *heat map* plot shows a two-way layout of the most differentially expressed genes along the vertical axis versus the experimental conditions on the horizontal axis. This graph is also hyper linked to the annotation information if it exists.

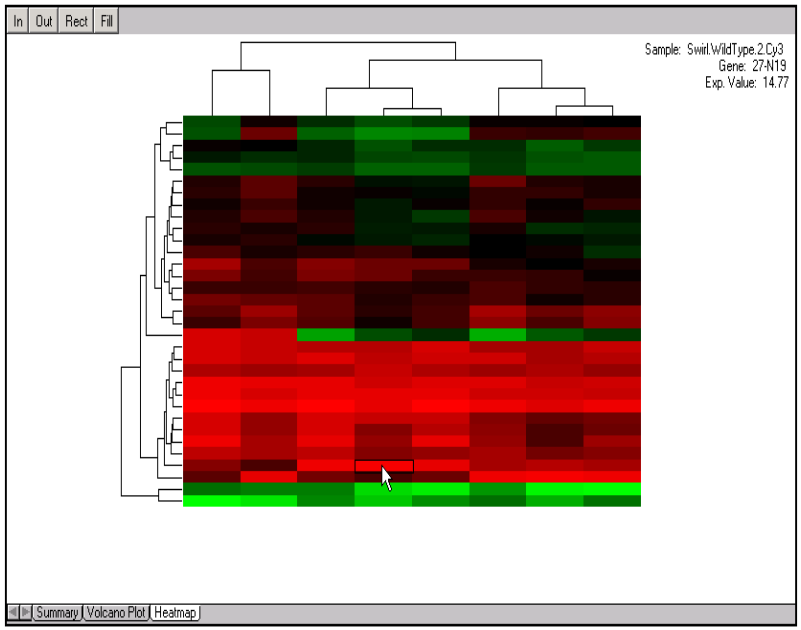


Figure 4.16: *A heat map plot shows differentially expressed genes as a function of experimental conditions.*

TWO-WAY REFERENCE DESIGN

Malaria Parasite Data

In this section, we examine two-color microarray data from another developmental biology experiment. The data come from the study of one stage of the life cycle of *Plasmodium falciparum*, one of four parasitic protozoa which cause human malaria which annually affects 200-300 million people worldwide. By understanding the life cycle through gene expression, "insight into the biochemical function and regulation of these genes will provide the foundation for future drug and vaccine development efforts toward eradication of [malaria]." (Bozdech, et. al., 2003)

The study was designed to track gene expression closely over a 48 hour period of time of a single stage of the life cycle. The data consists of replicate dye-swapped pairs of arrays taken at life-cycle time points 1, 7 11, 27 and 31 hours. Total RNA at each time point of the life cycle (LC) was compared to an arbitrary reference pool of RNA from all time points. See Bozdech, et. al. (2003) for more detail. We acknowledge Dr. Joseph DeRisi's generous offer to allow Insightful to include these data with S+ARRAYANALYZER. The complete data are also freely available for download from

```
http://dx.doi.org/10.1371/journal.pbio.0000005.sd001
```

The experimental design is a two-way reference design with observations at the design points displayed in Table 4.2.

Table 4.2: *The malaria parasite experimental design.*

C532	C635	Time	Array	File Name
Ref	LC	1	1	TP_01a.gpr
LC	Ref	1	2	TP_01b.gpr
Ref	LC	7	3	TP_07a.gpr
LC	Ref	7	4	TP_07b.gpr
Ref	LC	11	5	TP_11a.gpr

Table 4.2: *The malaria parasite experimental design.*

C532	C635	Time	Array	File Name
LC	Ref	11	6	TP_11b.gpr
Ref	LC	27	7	TP_27a.gpr
LC	Ref	27	8	TP_27b.gpr
Ref	LC	31	9	TP_31a.gpr
LC	Ref	31	10	TP_31b.gpr

The fundamental questions of the study were

1. *Which genes are active or expressing at any point in time, and*
2. *How did gene expression change over time?*

To answer Question 1, we need to compare the life-cycle samples to the reference samples at each time point. To answer Question 2, we need to compare the life cycle samples across time points.

Importing Data

Importing the malaria parasite data is similar to importing the mutant Zebrafish data that we did at the beginning of this chapter. The main difference is in the design setup and specifying variables for later filtering. We'll focus on these two aspects of the import process and refer you back to the beginning of the chapter for more complete information on the data import process.

Open the (Import Data) From Two Channel by clicking **ArrayAnalyzer ► Import Data ► From Two Channel**. Then open the **Create/Modify Design** dialog by clicking on the **Create/ Modify Design** button at the top of the dialog. Here we specify 10 arrays, two factors and factor names of **LCRef** with values of **LC** and **Ref** and **Time** with values of **1, 7, 11, 27** and **31**. The resulting dialog is displayed in Table 4.17

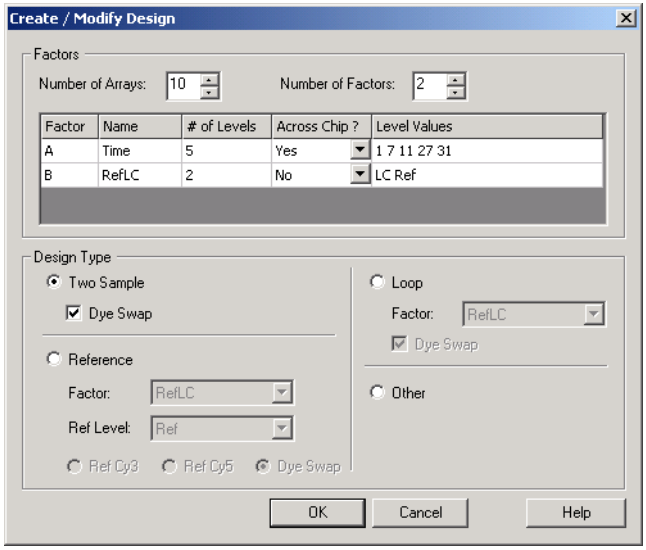


Figure 4.17: *The **Create/Modify Design** dialog for the Malaria Parasite data.*

Creating the Design

Now click OK on the Create/Modify Design dialog to copy the design into the file selection grid and associate the files which each design point. You can find the malaria parasite example data by navigating to your **splus62/module/ArrayAnalyzer/examples** directory, selecting **GPR** (***.gpr**) as the **Files of type**, and selecting **TP_01a.gpr**. Repeat for the other 10 **.gpr** files, entering one file per cell. Alternatively, read the design file, **TPDesignFile.txt** from the examples directory.

The Malaria Parasite data (TP files) are not installed by default.

You will only find the TP example files in the **examples** directory if you have done a Full install of S+ARRAYANALYZER. The TP dataset requires about 300MB of disk space.

Creating The Layout

The create the layout Click the **Create Array Layout** button and browse to the **examples** directory for **TPLayout.gal**. Selecting **Files of type: GAL** when you are browsing will make it easier to find **TPLayout.gal**. The other settings are as displayed in Figure 4.18.

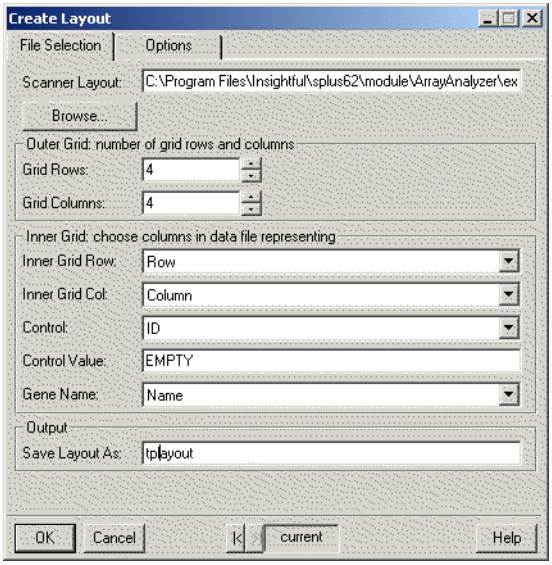


Figure 4.18: *The layout setup for the malaria data.*

Selecting Expression Variables

Now shift to the **Variable Selection & Filtering** tab and select **F532.Mean**, **B532.Mean**, **F635.Mean**, and **B635.Mean** for **Green Foreground**, **Green Background**, **Red Foreground**, and **Red Background**, respectively.

Selecting Filtering Variables

While still on the **Variable Selection & Filtering** tab, select the **Flags** variable in the **Extra Variables** (**for filtering later**) list and click on the >> button to move it into the **Keep** list on the right side of the dialog. The Flags column marks problem spots that should be filtered out before doing any serious analysis.

The result of selecting expression and filtering variables is displayed in Figure 4.19.

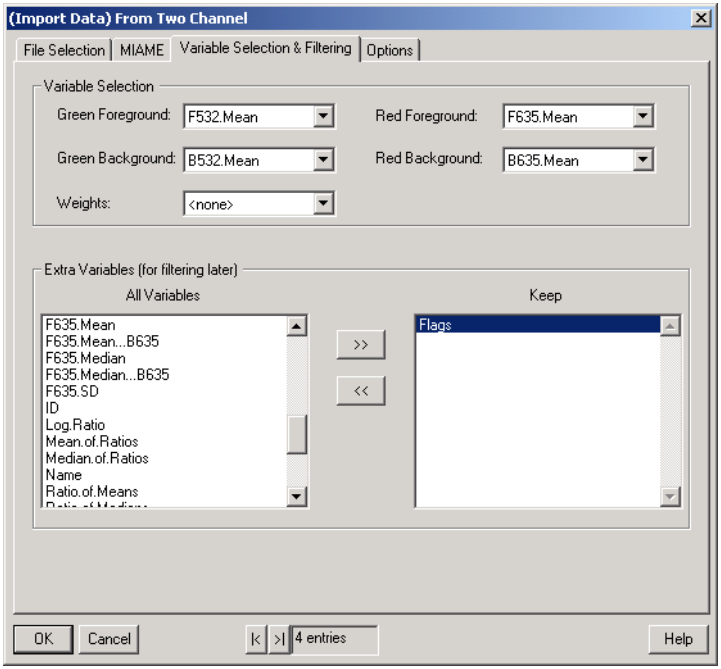


Figure 4.19: *Selecting variables for expression analysis and filtering.*

Quality Diagnostics

Once you've read the data in you can generate an assortment of visual quality checks on the data. These checks include:

- Image plots of the entire array for any of the channels. Multiple color schemes are available for the plots.
- MvA plots as scatter plots or hexbin plots. Hexbin plots display hexagonal "points" that are colored to give a sense of the *density* of intensity values at each location.
- Expression intensity boxplots by array or by print-tip group for each array.
- Principal components plot where points represent each array. For this plot, all the expression intensities for each array are considered the variables of interest.

For more information on the methods and interpretation of these visual quality diagnostics see Chapter 5, Quality Control Diagnostics and Filtering.

Select **ArrayAnalyzer ► Quality Control Diagnostics** and fill in the resulting dialog as displayed in Figure 4.20. Clicking **OK** or **Apply** generates the plots seen in Figures 4.21, 4.22, 4.23 and 4.24.

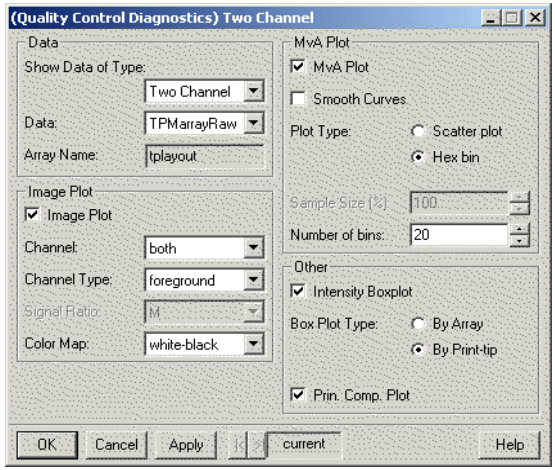


Figure 4.20: ***(Quality Control Diagnostics) Two Channel*** *dialog setup.*

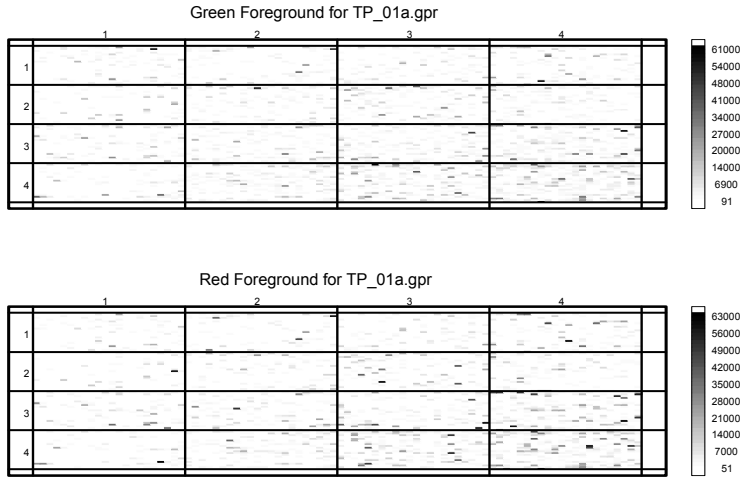


Figure 4.21: *Image plots of red and green foreground channels.*

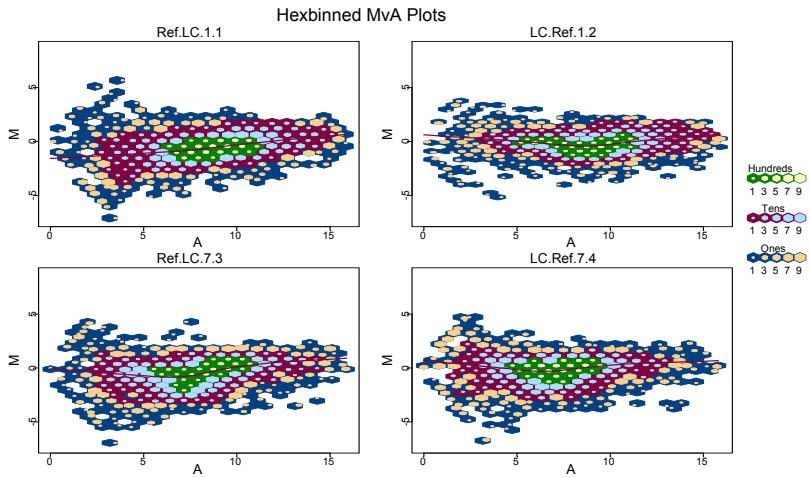


Figure 4.22: *Hexbinned MvA plots for four of the malaria arrays. The lines overlaid indicate locally weighted mean value with confidence bounds.*

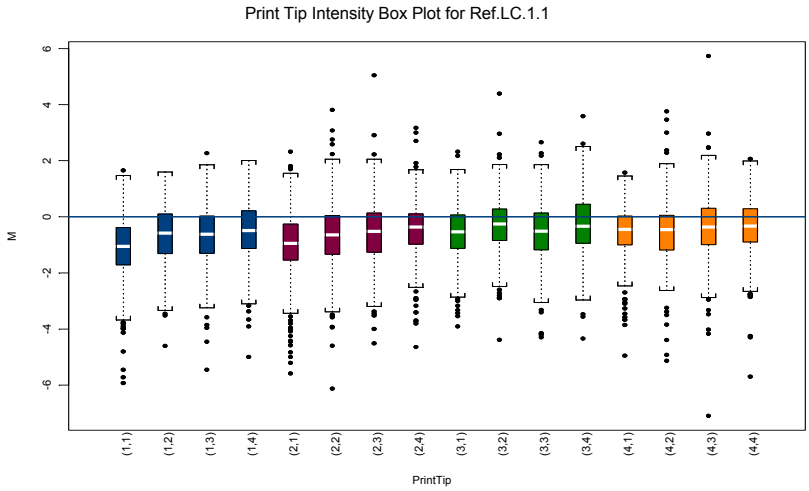


Figure 4.23: *Intensity boxplot by print-tip group for the malaria parasite data. Note the trend within each print-tip row indicating a need for applying a normalization method that remove this print-tip group bias.*

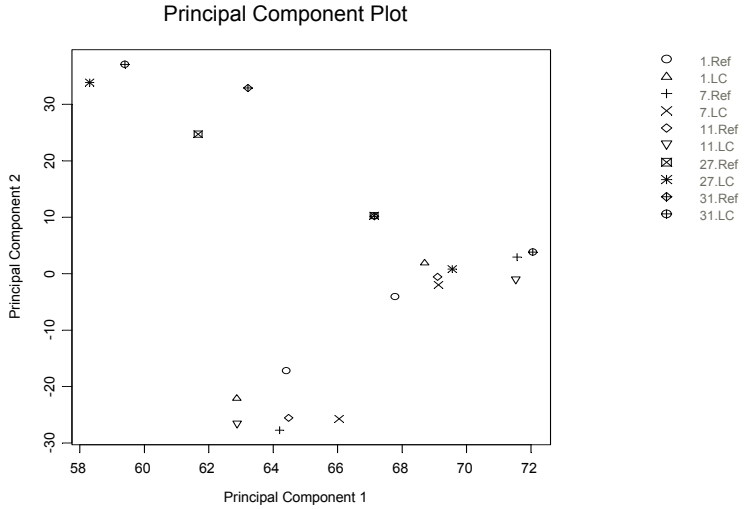


Figure 4.24: *Plot of first two principal components with taking all expression intensities for each array as the variables.*

Filtering

The QC plots indicate the need to normalize, but before we do that we should eliminate problem spots. We kept the **Flags** column for each of the arrays. A non-zero flag value indicates some problem occurred in calculating the intensity value or the spot was "EMPTY" during the experiment. We can eliminate these spots using the **Filtering** dialog. From the main S-PLUS menu bar click to the filtering dialog by doing **ArrayAnalyzer ▶ Filtering**.

The **Filtering** dialog has two pages, 1) **Array Filtering** for filtering out *entire arrays* and 2) **Gene Filtering** for filtering out *genes*. On the first tab select **Data of Type: Two Channel** and then select the **TPMarrayRaw** object for filtering. In the **Save As** field of the **Output** group type in **TPMarrayRawFiltered** to save the resulting object.

Leave the rest of the first tab alone because we don't want to eliminate any array in its entirety. Proceed to the **Gene Filtering** tab.

The **Gene Filtering** tab is designed to create filtering expressions for dropping or keeping genes (*rows* in the expression matrix). For this example, use the **Flags** column to create expressions for eliminating rows with non-zero flags. Do so in the following order:

1. Select a data column, **Flags**, and click the **Add** button in the Data group to add the name **Flags** to the **Expression** field at the bottom of the dialog.
2. Select the == symbol (double equal sign) from the **Logical** field of the **Operations** group and click the **Add** button right under **Logical** label to add the == symbol to the **Expression** field at the bottom of the dialog.
3. Select the value 0 (zero) from the list in the **Column Values** group and click the **Add** button in that group to add 0 (zero) to the **Expression** field at the bottom of the dialog.

4. Select 1 (one) in the **InAtLeast Value** field of the **Operations** group and click the **Add** button right under the **InAtLeast Value** label to add **InAtLeast 1** to the **Expression** field at the bottom of the dialog.

Note on InAtLeast

The **InAtLeast 1** operation allows you to keep any gene that has a **Flag** value of zero on one or more of the arrays. Without this operation a gene will be eliminated even if the Flag value is non-zero on only one array.

5. Ensure that the **Values Selected** in the **Expression** group is set to **Keep**.
6. Click **OK** to eliminate all genes except those with Flag == 0.

The resulting **Gene Filtering** tab of the **Filtering** dialog is displayed in Figure 4.25.

Note that the report generated by the filtering operation indicates the number of genes dropped.

```
Dropped: 461  Retained: 7283  Out of: 7744 Genes
```

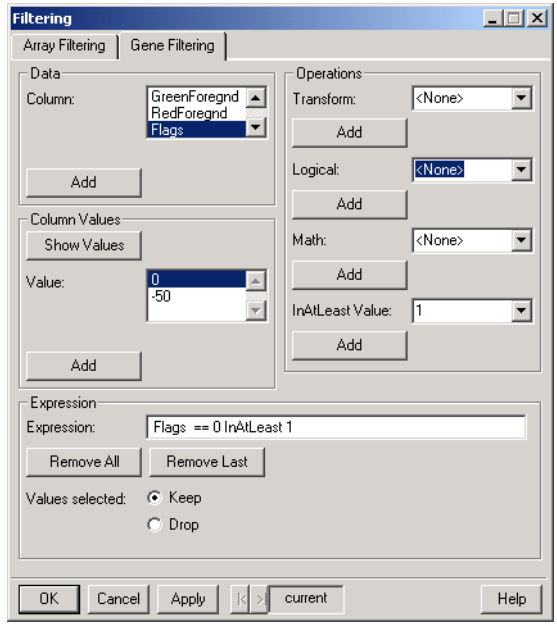


Figure 4.25: *Keep only the genes with zero flags.*

Quality Diagnostics Revisited

After the filtering operation another QC check shows regions of the arrays that had problem or missing values, i.e., flags that were set to non-zero. To see this go back to **Quality Control Diagnostics ▶ Two Channel** and run the image diagnostics again for the filtered object **TPMarrayRawFiltered**. Select the **Image Plot** checkbox, **both** for the **Channel** and **foreground** for **Channel Type**. By choosing a **Color Map** which has non-white low values you can see the regions that have been eliminated across the arrays. Figure 4.26 displays one of the post-filtering image plots. The white strips in the bottom right corner of each print-tip group correspond to empty spots that occurred across all arrays.

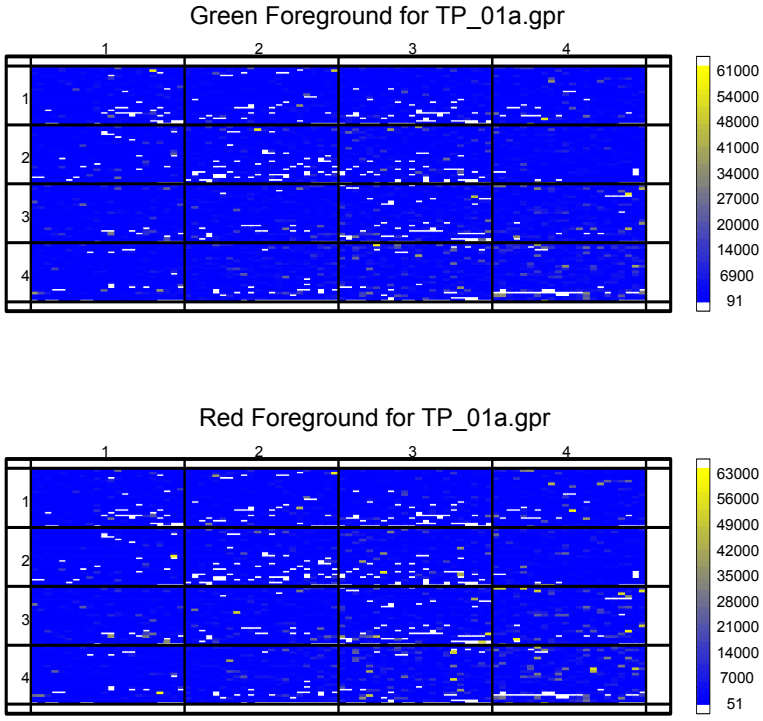


Figure 4.26: *Image plots of both foreground channels after filtering. Note the white strips at the bottom right of each of the print-tip groups which were non-zero flags for EMPTY spots.*

Normalization

Now it's time to normalize the arrays. The boxplots in Figure 4.23 clearly show a dependency of expression intensity on print-tip group. In fact, there is even an indication of trending with increasing column number within any print-tip row. This indicates a need to apply normalization within the print-tip groups. We can accomplish this with one of the print-tip normalization methods available from the **Normalization** dialog.

Click **ArrayAnalyzer ► Normalization** from the main S-PLUS menu bar. In the resulting dialog select **Two Channel** data in the **Show Data of Type** field and select the **TPMarrayRawFiltered** object for normalization. Note that the Save As field is automatically filled with the object name **TPMarrayRawFiltered.norm**.

Now choose the **scalePrintTipMAD** method in the **Normalization** field of the **Normalization** group on the right side of the dialog. This method effectively equates the median and variance of each print-tip group within each array. We won't normalize across arrays because we will adjust for array error (a block effect) in the differential expression analysis step. The resulting dialog settings are displayed inFigure 4.27.

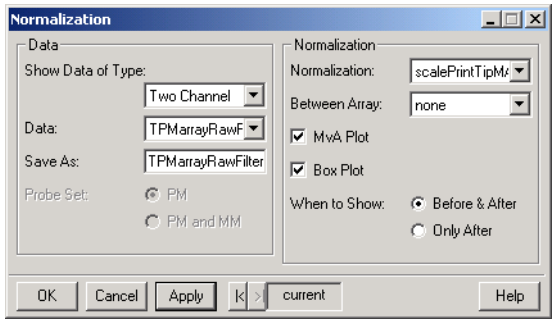


Figure 4.27: *Normalization dialog settings for the filtered malaria data.*

Click OK or Apply to run the normalization procedure. The normalization step produces the before and after boxplots of $\log_2$ intensity ratios as displayed in Figure 4.28. This plot displays each array as a separate boxplot. Note the alignment of the medians and some normalization of the variance across arrays even though normalization was done completely within arrays.

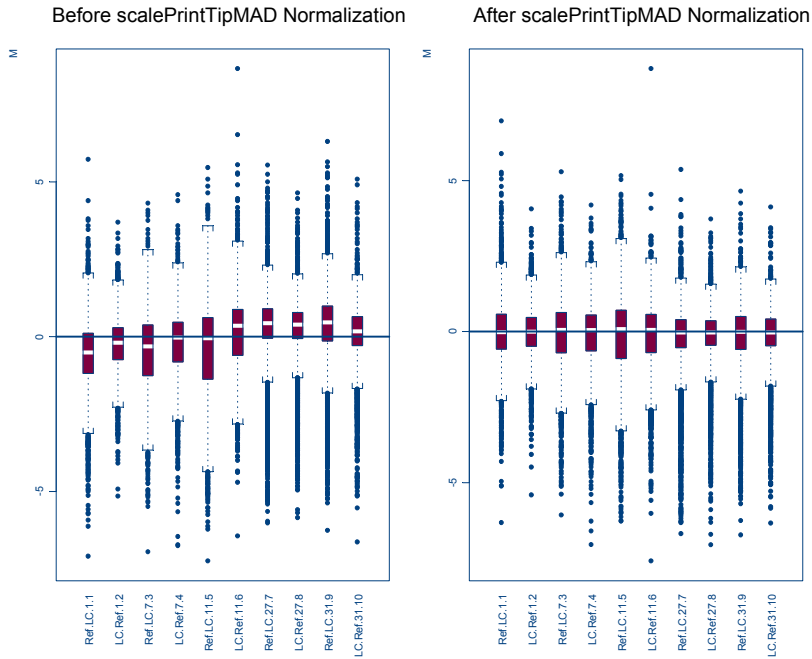


Figure 4.28: *Intensity boxplots before and after normalization.*

You can check the result of the normalization step by going back to the **Quality Control Diagnostics** for **Two Channel** dialog and re-running the print-tip group boxplots for the normalized data in **TPMarrayRawFiltered.norm**. Figure 4.29 displays the print-tip group boxplots for the same array as in Figure 4.23. Note the removal of the trend we see in the *un-normalized* plot and the alignment of medians and shoulders of the boxes. The **scalePrintTipMAD** method is effective in removing the systematic sampling effects which produced the print-tip group biases seen in Figure 4.23.

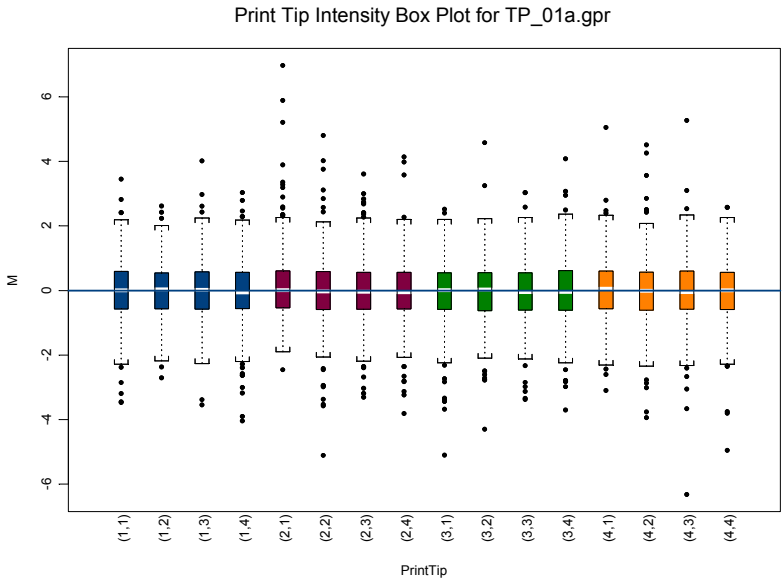


Figure 4.29: *Boxplots by print-tip group after* ***scalePrintTipMAD*** *normalization. The normalization has been effective in removing the systematic sampling effects seen in the non-normalized data.*

Clustering Expression

The analysis of (Bozdech, et. al., 2003) reveals changing gene expression over the course of the study. Groups of genes expressing early in the study no longer do at the end. Recall that times ranged from 1 to 48 hours in the original study. Our example data set ranges from 1 to 31 hours. With a simple hierarchical clustering of expression intensities we can examine the change in expression over time.

Open the Clustering dialog by clicking **ArrayAnalyzer ► Cluster Analysis** from the main S-PLUS menu bar. On the **General Options** tab, select **Two Channel** from the **Show Data of Type** drop-down list and **TPMarrayRawFiltered.norm** from the **Data** drop-down list. Leave the other fields are they are by default and switch to the **Filtering Options** tab.

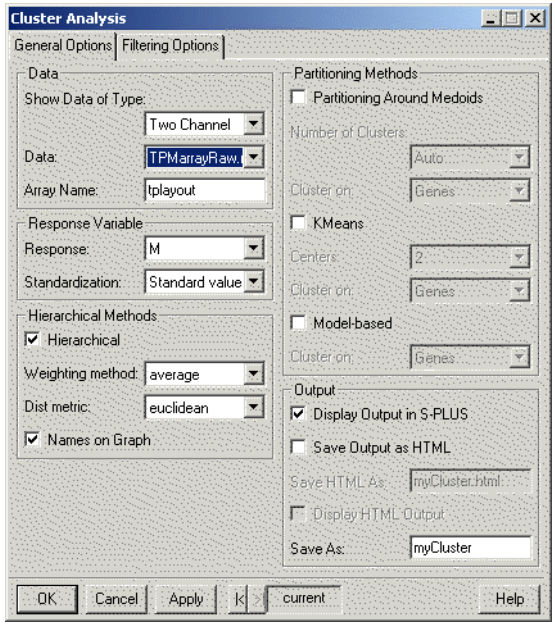


Figure 4.30: ***General Options*** *tab of the Cluster Analysis dialog ready for a hierarchical cluster analysis of the TP data.*

On the Filtering Options tab in the Gene Sort Order Options group on the far right, change the **Limit number of genes to: 500**. This will allow us to cluster on the 500 most expressive genes.

The two tabs should now look like Figures 4.30 and 4.31.

Now click **OK** or **Apply** to run the analysis.

Figure 4.32 displays the results of the cluster analysis. The dendrogram on the vertical axis corresponds to clustering on genes. The dendrogram on the (top) horizontal axis corresponds to clustering on experimental conditions or arrays. In this graph the early times (1, 7, 11) are on the right side (the right six channels) and the later times (27, 31) are on the left (left four channels). Note the clear separation in expression between the two time groups. Genes that are positively expressing (green in color) during the early times are negatively expressing (red in color) during the late times. In fact, if you look closely at the expression pattern in the right six channels you can see a shift in the expression values for genes in the top of the list as you scan from 1 hour to 11 hours. The values shift from

negative (red) for 1 hour to positive (green) at 11 hours.This provides a simple verification of the expression patterns that (Bozdech, et. al., 2003) discovered in their analysis.

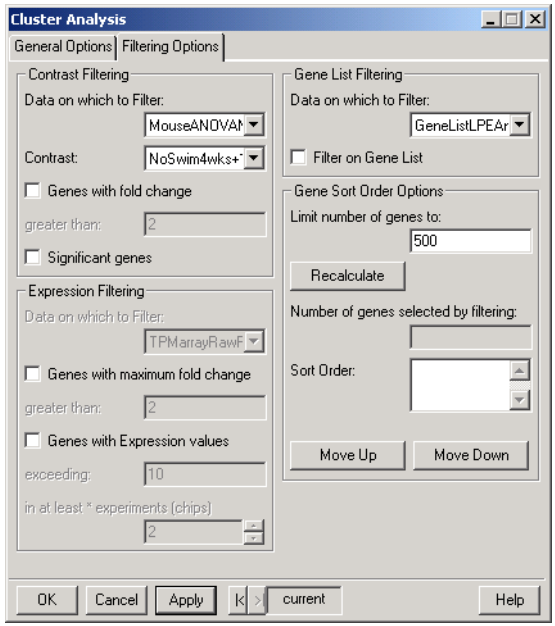


Figure 4.31: ***Filtering Options*** *tab of the* ***Cluster Analysis*** *dialog ready for a hierarchical cluster analysis of the TP data*

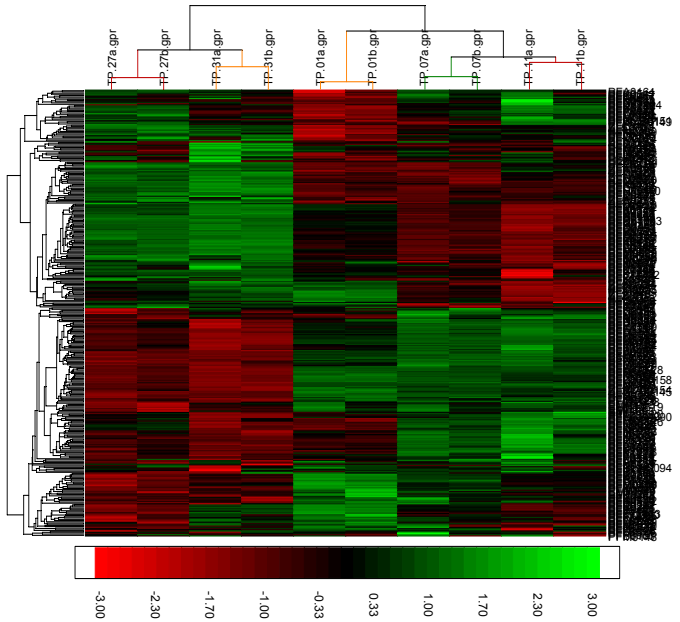


Figure 4.32: *Hierarchical clustering of the 500 most expressive genes in the TP data. Note the dramatic shift in expression values between late (left four channels) and early (right six channels) times points.*

Differential Expression Analysis

We are now ready to focus on differential expression analysis. Recall, when we set up the design we stated one of the critical questions was to find how gene expression changed over time. We can test for that by looking a simple contrasts between time points in the study.

To set up the analysis click **ArrayAnalyzer ▶ Differential Expression Analysis ▶ LPE Test**. Select **Two Channel** data type, **TPMarrayRawFiltered.norm** as the **Data** drop-down list and **BH** for adjustment. Run a contrast comparison by selecting Time as the Factor and 31 as Compare Level 1 and 1 as Compare Level 2. Name the **Save As** object **TPLPET31m1**. The resulting dialog is displayed in Figure 4.33. Repeat the analyses for time points 31 and 7, and 31 and 27 naming the result objects **TPLPET31m7** and **TPLPET31m27**, respectively. For time points with a large spread we would expect many differentially expressed genes given the results of the clustering. For closer time points we would expect fewer. We will examine these results with the Gene List Management dialog.

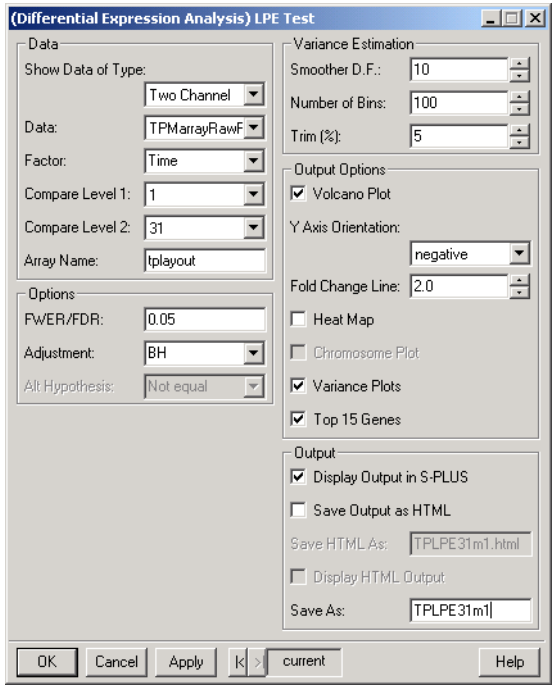


Figure 4.33: ***LPE Test*** *dialog set up for comparing time points 31 and 1.*

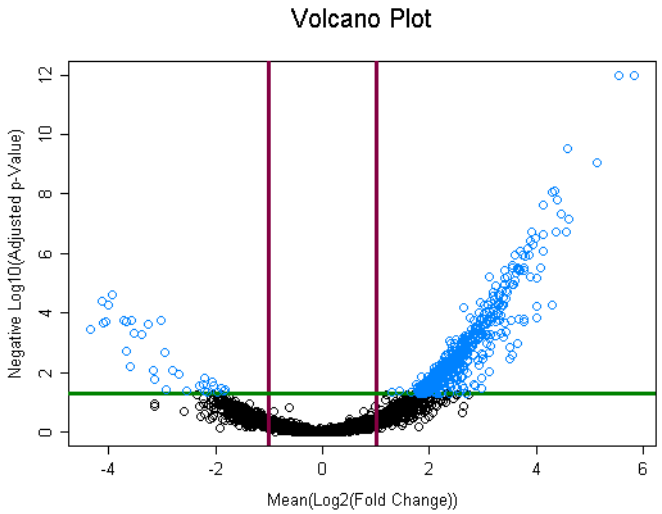


Figure 4.34: *Volcano plot for comparing time points 31 and 1.*

Gene List Management

The **Gene List Management** dialog allows you to merge and compare gene lists from testing different contrasts in an analysis. Open the dialog by clicking **ArrayAnalyzer ► Gene List Management**. Now in each of the three data groups select **DiffExprTest** for the **Data Type** and each of the contrast objects you created for comparing time points. The resulting dialog is displayed in Figure 4.35. For more information on the Gene List Management dialog see Chapter 9, Annotation and Gene List Management.

Clicking OK or Apply generates the Venn diagram displayed in Chapter 4.36, Examples: Two-Color Data. Note the large number of differentially expressed genes for large time spans and relatively few for short time spans. We expect short time spans to have similar gene lists and consequently few differentially expressed genes. Also note, little overlap between the three gene lists indicating changing gene expression over time.

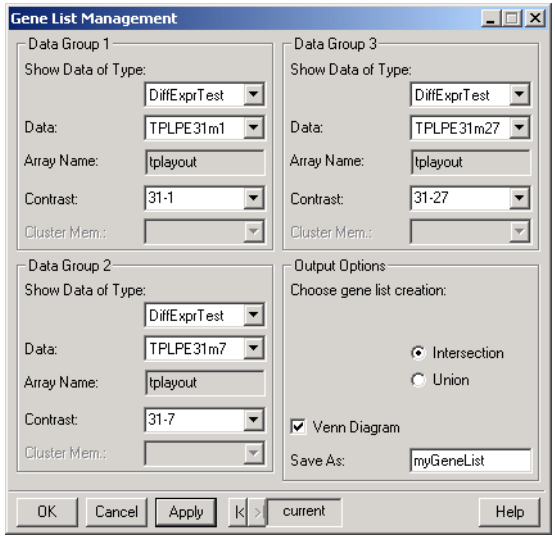


Figure 4.35: *The Gene List Management dialog set to compare gene lists from different contrast tests.*

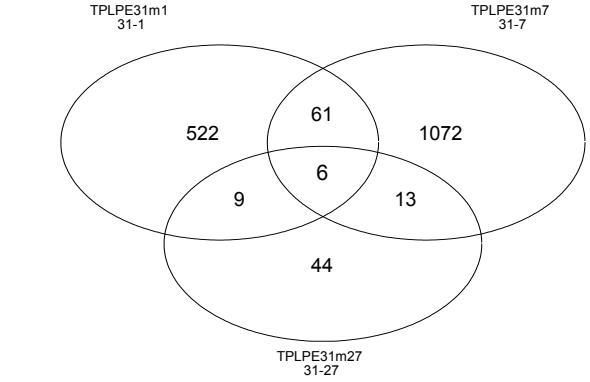


Figure 4.36: *Venn diagram contrasting lists generated by different contrast tests.*

Annotation

For Affymetrix Chips annotation is readily available on public databases (see Chapter 9, Annotation and Gene List Management for more detail) so we have automated annotation through interactive S-PLUS graphlets and the GUI. For examples see section Graphical Annotation on page 46 or section Annotation on page 50. We can set up graphical annotation for any two-channel array as long as the annotation information is available via a typical http (internet) protocol and the URL's to individual gene annotation are systematically structured. The URL structure required is a base URL plus a gene ID which when concatenated together creates the unique URL to the specific gene annotation page.

The TP is associated with a web site dedicated to Malaria parasite annotation with appropriate URL naming structure so we'll use it as an example of setting up annotation for custom (non-standard) arrays. To create graphical annotation for the TP test results proceed as follows:

1. Create an (S-PLUS) *named list* containing all the gene ID's which are part of the gene-specific URL.
2. Create a small data frame with the following information:

   ANNODATA: character string which is appended onto the chip name to create the name of the named list created in 1)

   COLNAMES: column name of gene names in the layout file.

   URL: base URL to the annotation database. Gene ID must be appended to complete the URL.

   MENULABEL: label displayed in the drop-down menu of the graphlet.

   DISPLAYNAME: column name used on HTML gene list output.

Creating the Named List

The easiest way to do this is to read in the layout file in a separate S-PLUS data frame. You can do that either through the command line or through the GUI via the S-PLUS menu bar. We demonstrate the example through the GUI.

- Select **File ► Import Data ► From File** to open S-PLUS' **Import From File** dialog.

- Browse to the **TPLayout.gal** file in the **examples** directory for S+ARRAYANALYZER and select it.
- Set the **File Format** field to **ASCII File - whitespace delim** (**asc;dat;txt;prn**).
- Type in the **Data set** name, **TPLayoutComplete**, in the **To**.
- On the **Options** tab
  - type in **6** for **Col names row**,
  - **6** for **Start row** to skip the header rows, and
  - deselect the **Strings as factors** checkbox.
- Click **OK** to read in the **GAL** file.

The completed Data Specs and Options tabs are displayed in Figures 4.37 and 4.38.

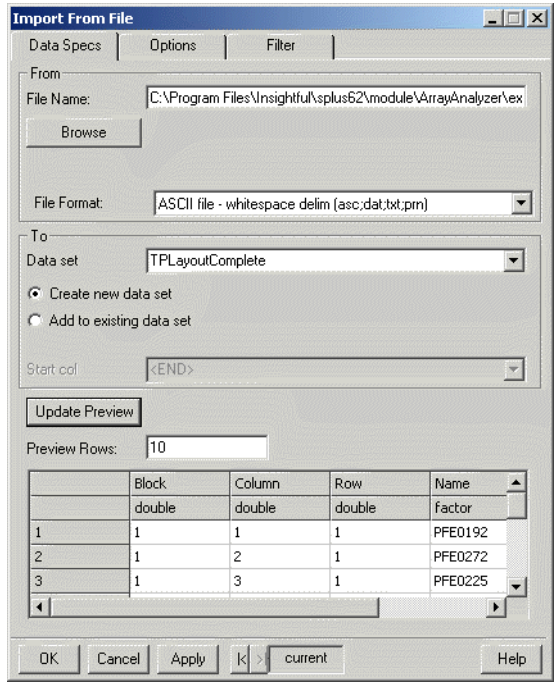


Figure 4.37: ***Data specs*** *for importing gene ID data.*

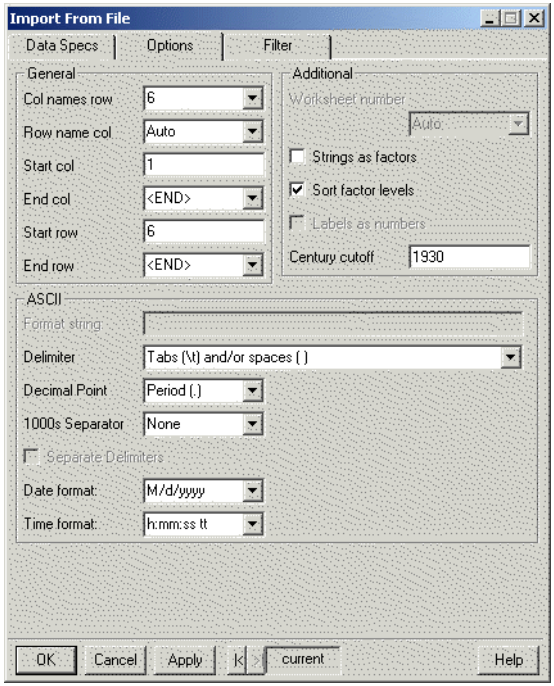


Figure 4.38: ***Options*** *for importing gene ID data.*

Now from the S-PLUS command line create the named list as follows:

```
tplayoutPF <- as.list(TPLayoutComplete$ID)

names(tplayoutPF) <- TPLayoutComplete$Name
```

tplayoutPF is the named list with gene ID's used to construct the URL's for annotation.

Note
The first part of the name of the named list, **tplayout**, *must* match the name of the layout object you created when you read in the data. Also the layout object name *must be all lower case.*

Creating the Information Data Frame

To create the data frame that is used by the graphlet for type the following command into the command line window.

```
AA.Annotation.Columns <- data.frame(
```

```
ANNODATA=c("PF"),
COLNAMES=c("PF.ID"),
URLS=c("http://malaria.ucsf.edu/ligolink.php?OLIGO="),
MENULABEL=c("PF ID"),
DISPLAYNAME=c("PF.ID"),
stringsAsFactors=F)
```

Now we are ready to re-run the analysis described in section Differential Expression Analysis on page 164. The significant genes in the summary table and the volcano plot are hyper-linked to the DeRisi Lab Malaria Transcriptome Database. Figures 4.39, 4.40 and 4.41 display the results.

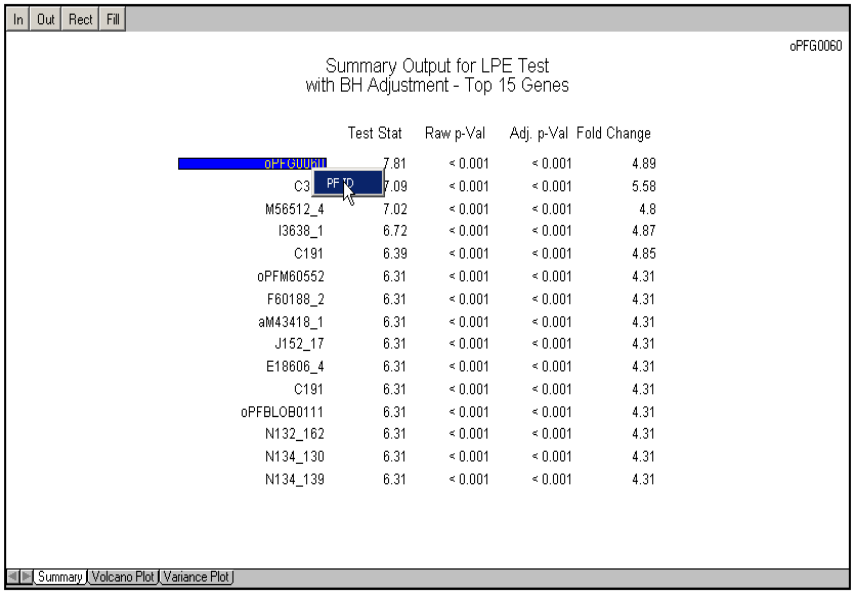


Figure 4.39: *Hyperlinking from the Top 15 Gene List to the DeRisi Lab Malaria Transcriptome Database.*

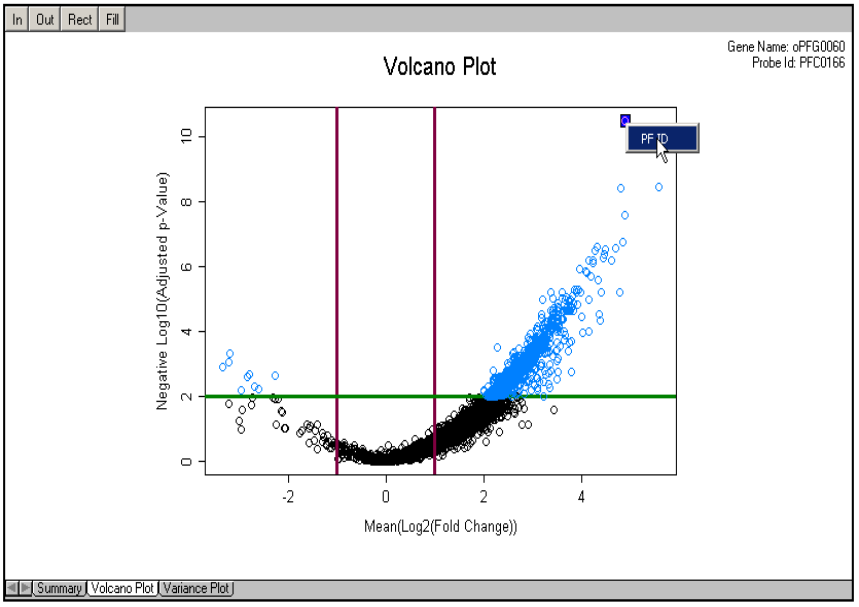


Figure 4.40: *Hyperlinking from the Volcano Plot to the DeRisi Lab Malaria Transcriptome Database.*

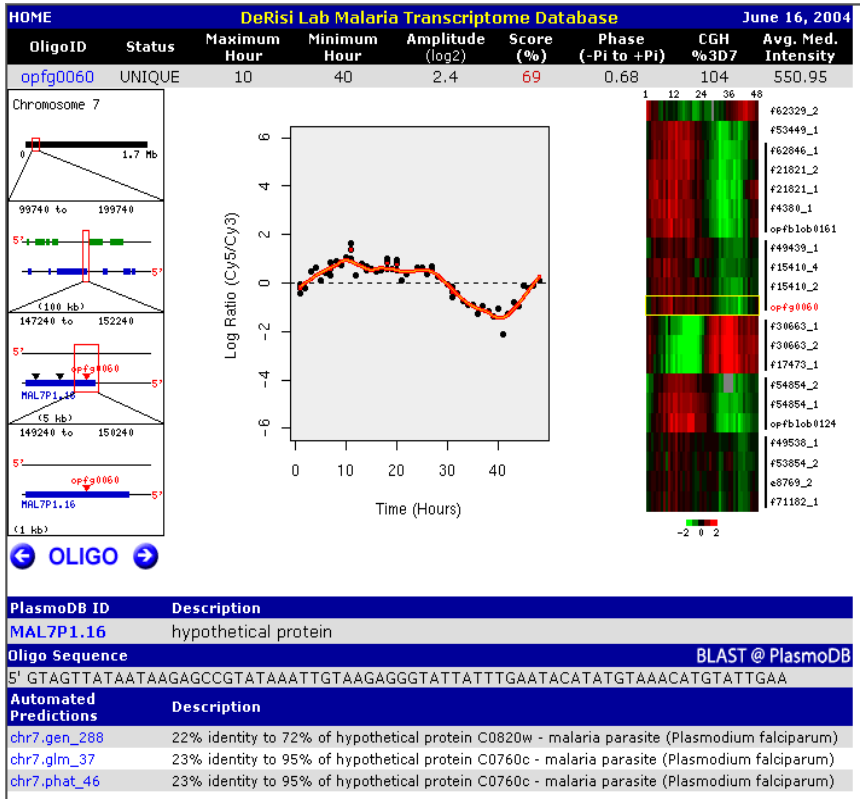


Figure 4.41: *Annotation page from the DeRisi Lab Malaria Transcriptome Database.*

FROM THE COMMAND LINE

All of the analysis done through the GUI can be done from the S-PLUS command line. Having access to the command line adds great flexibility to the set of features available through the ArrayAnalyzer GUI and opens the door to additional analyses. The flexibility and feature-rich S-PLUS language make it an ideal platform for exploratory analysis, statistical testing and modeling of gene expression data.

This section is designed to expose you to the critical functions for differential expression testing of microarray data. If you do not plan to run your analyses from the command line, you can skip this section.

Importing Data

The relevant information for a cDNA microarray is

1. layout of the chip
2. experimental design
3. gene ID's
4. expression intensities

All of this information must be read into S-PLUS and assembled into a single object for further analysis. The primary convenience of importing data through the GUI is the coordination of the following three functions which read the above information:

`read.marrayLayout` This function creates objects of class `marrayLayout` to store layout parameters for two-color cDNA microarrays.

`read.marrayInfo` This function creates objects of class `marrayInfo`. The `marrayInfo` class is used to store information regarding the target mRNA samples co-hybridized on the arrays or the spotted probe sequences (e.g. data frame of gene names, annotations, and other identifiers).

`read.marrayRaw` This function reads in cDNA microarray data from a directory and creates objects of class `marrayRaw` from spot quantification data files obtained from image analysis software or databases.

We'll step through an example using each of the functions to give you a flavor of their use. They are listed in typical order of use.

Reading Layout Information

Layout files describe the structure of the microarray. They include information on the arrangement of the spots (e.g., number of rows and columns) on the array, where each gene is located, which spots are control spots, etc. A typical layout file is the file, **fish.gal**, provided as an example for the swirl data. It has 21 lines of header information and then starts a data table. The file is located in the **examples** directory of the ArrayAnalyzer module. You can find the location of the file by doing:

```
> AApath <- file.path(getenv("SHOME"), "module",
            "ArrayAnalyzer", "examples")
> AApath
"C:/PROGRAM FILES/INSIGHTFUL/
splus62\\module\\ArrayAnalyzer\\examples"
```

By scanning the file in Notepad or Wordpad you can see that the spots are arranged in 16 blocks (a 4 x 4 grid) of 22 rows and 24 columns. Note that the `ID` column (column 4) has indicates which spots are controls. We are now ready to read in the layout file.

```
> swirl.layout <-
   read.marrayLayout(file.path(AApath,"fish.gal"), ngr = 4,
   ngc = 4, nsr = 22, nsc = 24, skip = 21, ctl.col = 4)

> swirl.layout
Array layout: Object of class marrayLayout.

Total number of spots:  8448
Dimensions of grid matrix:  4 rows by 4 cols
Dimensions of spot matrices:  22 rows by 24 cols

Currently working with a subset of 8448 spots.

Control spots:
There are   7681 types of controls :
 control fb16a01 fb16a02 fb16a03 fb16a04 fb16a05 fb16a06
     768       1       1       1       1       1       1
       ...

Notes on layout:
```

```
C:/PROGRAM FILES/INSIGHTFUL/
splus62\module\ArrayAnalyzer\examples\fish.gal
```

Note that the column that had control indicators also had gene ID's we need to correct that in the `swirl.layout` object. We do that by using a couple of utility functions, `maNspots` and `maControls`.

```
> controls <- rep("control", maNspots(swirl.layout))
> controls[maControls(swirl.layout) != "control"] <- "N"
> maControls(swirl.layout) <- factor(controls)
> swirl.layout
Array layout: Object of class marrayLayout.

Total number of spots:8448
Dimensions of grid matrix:4 rows by 4 cols
Dimensions of spot matrices:22 rows by 24 cols

Currently working with a subset of 8448 spots.

Control spots:
There are   2 types of controls :
 Control    N
     768 7680

Notes on layout:
C:/PROGRAM FILES/INSIGHTFUL/
splus62\module\ArrayAnalyzer\examples\
fish.gal
```

The functions, `maControls` and `maNspots`, return the control vector and the number of spots, respectively, for an object of class `marrayLayout`.

Reading Experiment Information

This step reads the file with the experimental design information. For the example shown the informations is contained in the file `SwirlSample.txt` located in the `examples` directory of the `ArrayAnalyzer` module.

```
> swirl.samples <- read.marrayInfo(file.path(AApath,
                     "SwirlSample.txt"))
```

The resulting object show the information stored in a rectangular array.

```
> swirl.samples
```

```
Object of class marrayInfo.

  maLabels # of slide        Names experiment Cy3
1       81         81 swirl.1.spot          swirl
2       82         82 swirl.2.spot      wild type
3       93         93 swirl.3.spot          swirl
4       94         94 swirl.4.spot      wild type

  experiment Cy5      date comments
1      wild type 2001/9/20       NA
2          swirl 2001/9/20       NA
3      wild type 2001/11/8       NA
4          swirl 2001/11/8       NA

Number of labels:  4
Dimensions of maInfo matrix:  4  rows by  6  columns

Notes:
C:/PROGRAM FILES/INSIGHTFUL/
splus62\module\ArrayAnalyzer\examples\SwirlSample.txt
```

Reading Gene ID's We are now ready to read the gene ID's.

```
> swirl.gnames <- read.marrayInfo(file.path(AApath,
   "fish.gal"),info.id = 4:5, labels = 5, skip = 21)
> swirl.gnames
Object of class marrayInfo.

   maLabels      ID  Name
 1    geno1 control geno1
 2    geno2 control geno2
 3    geno3 control geno3
 4    3XSSC control 3XSSC
 5    3XSSC control 3XSSC
 6     EST1 control  EST1
 7    geno1 control geno1
 8    geno2 control geno2
 9    geno3 control geno3
10    3XSSC control 3XSSC
...

Number of labels:  8448
```

```
Dimensions of maInfo matrix:  8448  rows by  2  columns

Notes:
C:/PROGRAM FILES/INSIGHTFUL/
splus62\module\ArrayAnalyzer\examples\fish.gal
```

The additional arguments to `read.marrayInfo` are

`info.id` gene ID numbers and associated labels

`labels` the column number in `fname` which contains the names that the user would like to use to label spots or arrays (e.g. for default titles in `maPlot`

`skip` the number of lines of the data file to skip before beginning to read data.

Reading Expression Intensity Data

We can now read in the data. The four swirl files are located in the **examples** directory of the `ArrayAnalyzer` module like the other file read in the previous section. The function `read.marrayRaw` takes the following arguments:

fnames a vector of character strings containing the file names of each spot quantification data file. These typically end in **.spot** for the software Spot or **.gpr** for the software GenePix.

`path` a character string representing the data directory. By default this is set to the current working directory ("."). In the case where `fnames` contains the full path name, path should be set to NULL.

`name.Gf` character string for the column header for green foreground intensities.

`name.Gb` character string for the column header for green background intensities.

`name.Rf` character string for the column header for red foreground intensities.

`name.Rb` character string for the column header for red background intensities.

`name.W` character string for the column header for spot quality weights.

`layout` object of class `marrayLayout`, containing microarray layout parameters.

`gnames` object of class `marrayInfo` containing probe sequence information.

`targets` object of class `marrayInfo` containing target sample information.

Note that besides file names and location and the columns indicating foreground and background intensities, we need to supply the objects we just created, `swirl.layout`, `swirl.gnames` and `swirl.samples`. The command line call for reading the four data files is as follows:

```
> fnames <- paste("swirl", 1:4, "spot", sep = ".")
> swirl.raw <- read.marrayRaw(fnames, path = AApath,
   name.Gf = "Gmean", name.Gb = "morphG",
   name.Rf = "Rmean", name.Rb = "morphR",
   layout = swirl.layout, gnames = swirl.gnames,
   targets = swirl.samples)
> swirl.raw
Pre-normalization intensity data: Object of class marray
Raw.

Number of arrays: 4 arrays.

A) Layout of spots on the array:
Array layout: Object of class marrayLayout.

Total number of spots:8448
Dimensions of grid matrix:4 rows by 4 cols
Dimensions of spot matrices:22 rows by 24 cols

Currently working with a subset of 8448 spots.

Control spots:
There are   2 types of controls :
 Control    N
     768 7680


Notes on layout:
C:/PROGRAM FILES/INSIGHTFUL/
splus62\module\ArrayAnalyzer\examples\fish.gal

B) Samples hybridized to the array:
```

```
Object of class marrayInfo.

  maLabels # of slide        Names experiment Cy3
1       81         81 swirl.1.spot          swirl
2       82         82 swirl.2.spot      wild type
3       93         93 swirl.3.spot          swirl
4       94         94 swirl.4.spot      wild type

  experiment Cy5      date comments
1      wild type 2001/9/20       NA
2          swirl 2001/9/20       NA
3      wild type 2001/11/8       NA
4          swirl 2001/11/8       NA

Number of labels:  4
Dimensions of maInfo matrix:  4  rows by  6  columns

Notes:
C:/PROGRAM FILES/INSIGHTFUL/
splus62\module\ArrayAnalyzer\examples\SwirlSample.txt

C) Summary statistics for log-ratio distribution:
              Min. 1st Qu. Median  Mean 3rd Qu. Max.
swirl.1.spot -2.74   -0.79  -0.58 -0.48   -0.29 4.42
swirl.2.spot -2.72   -0.15   0.03  0.03    0.21 2.35
swirl.3.spot -2.29   -0.75  -0.46 -0.42   -0.12 2.65
swirl.4.spot -3.21   -0.46  -0.26 -0.27   -0.06 2.90

D) Notes on intensity data: Normalization
```

Filtering Out Controls

We can extract the controls with the subsetting method for `marrayRaw` objects as follows:

```
### Extract controls
> swirl.raw.controls <- swirl.raw[controls == "control",]
> swirl.raw <- swirl.raw[controls != "control",]
```

Quality Diagnostics

Comparative plots are produced with maPlot and maBoxplot. We can create the plots either within print-groups for a single chip or for all chips, disregarding print-tip groups.

```
### Boxplots of controls vs noncontrols by print-tip group
> graphsheet()
> par(mfrow = c(1,2))
> maBoxplot(swirl.raw.controls[, 3],
      main = "Controls by Print-Tip Group", srt = 90)
> maBoxplot(swirl.raw[, 3],
      main = "Non-controls by Print-Tip Group", srt = 90)
```

Figure 4.42 displays the resulting graph.

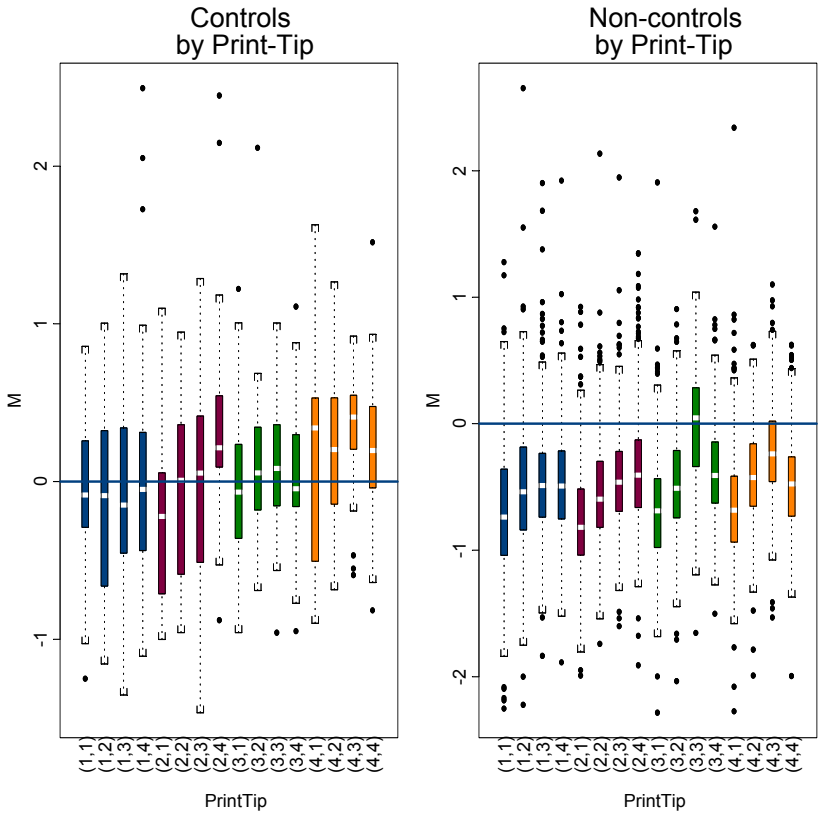


Figure 4.42: *Controls versus noncontrols by print-tip group.*

We can also plot controls versus noncontrols with a boxplot for each chip as follows:

```
### Boxplots of controls vs noncontrols
> graphsheet()
> par(mfrow = c(1,2))
> maBoxplot(swirl.raw.controls,
      main = "Controls by Print-Tip Group", srt = 90)
> maBoxplot(swirl.raw,
      main = "Non-controls by Print-Tip Group", srt = 90)
```

Figure 4.43 displays the resulting graph.

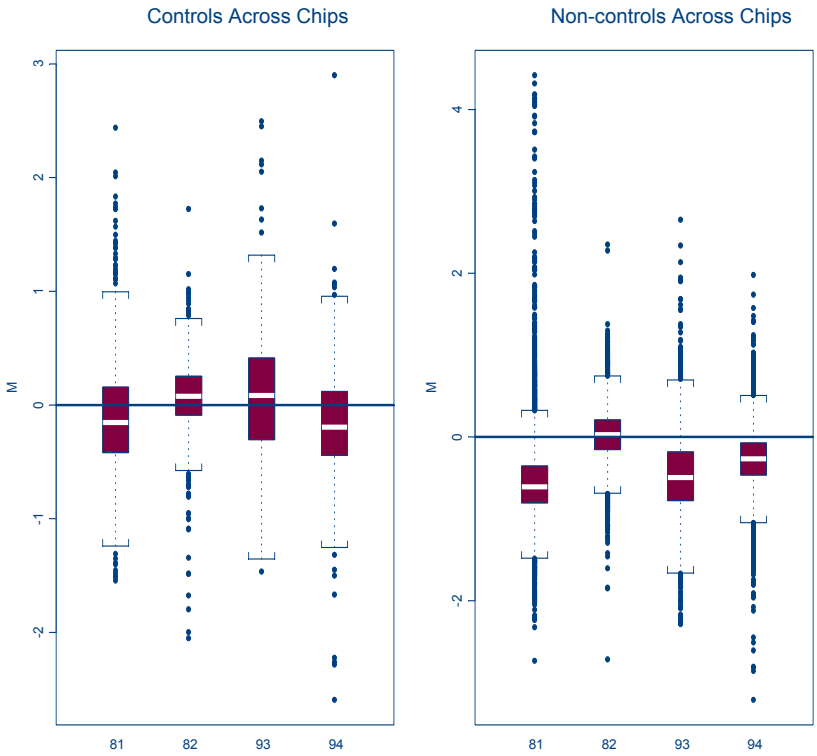


Figure 4.43: *Controls versus noncontrols across chips.*

Normalization

The `swirl.raw` object resulting from the call to `read.marrayRaw` is an object of class `marrayRaw`. Removing the controls does not effect the class of the object. The normalization function we use for `marrayRaw` objects is `maNorm`. Its arguments are listed here:

`mbatch` Object of class `marrayRaw`, containing intensity data for the batch of arrays to be normalized. An object of class `marrayNorm` may also be passed if normalization is performed in several steps.

`norm` Character string specifying the normalization procedures. The options to the `norm` argument are:

- `none` no normalization
- `median` global median location normalization
- `loess` global intensity or A-dependent location normalization using the `loess` function

`twoD` 2D spatial location normalization using the `loess` function

`printTipLoess` within-print-tip-group intensity dependent location normalization using the `loess` function

`scalePrintTipMAD` within-print-tip-group intensity dependent location normalization followed by within-print-tip-group scale normalization using the median absolute deviation (MAD).

The default normalization method is `printTipLoess`. To normalize `swirl.raw` with the `scalePrintTipMAD` method the function call is

```
> swirl.norm <-  maNorm(swirl.raw, norm = "s")
```

Note that only the first letter of the method, "`s`" in this case, is needed.

We can do a series of plots to compare before and after normalization First we do a pair of M vs A plots. The `maPlot` function handles all the details. We pick off one of the arrays (the third one) for simplicity.

```
> maPlot(swirl.raw[,3], main ="Pre-normalization MvA Plot")
> maPlot(swirl.norm[,3], main ="Post-normalization MvA")
```

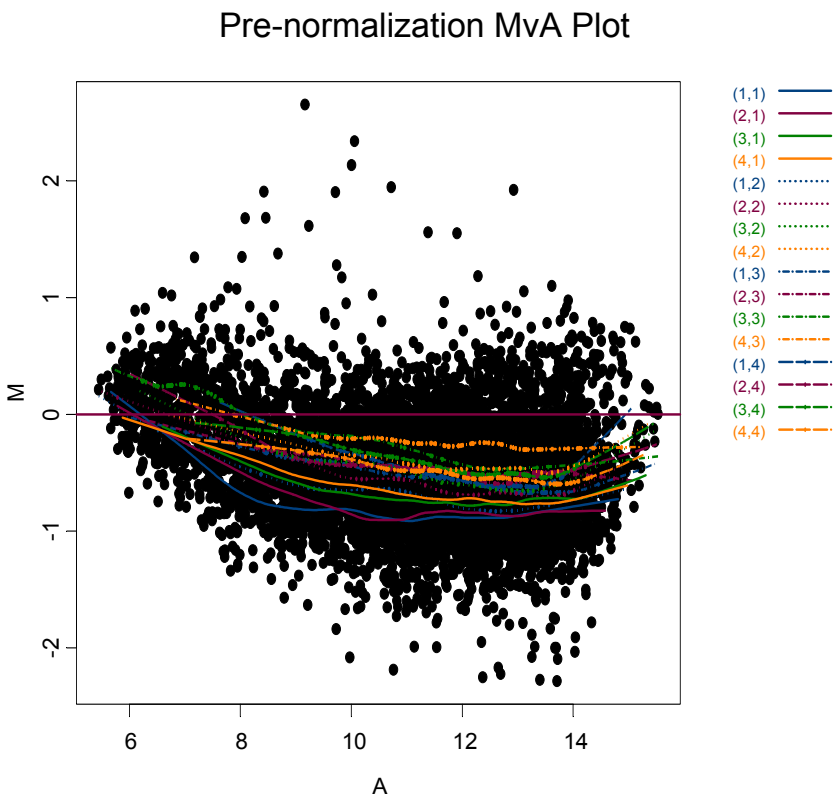


Figure 4.44: *Pre-normalized M vs A plot for the swirl data.*

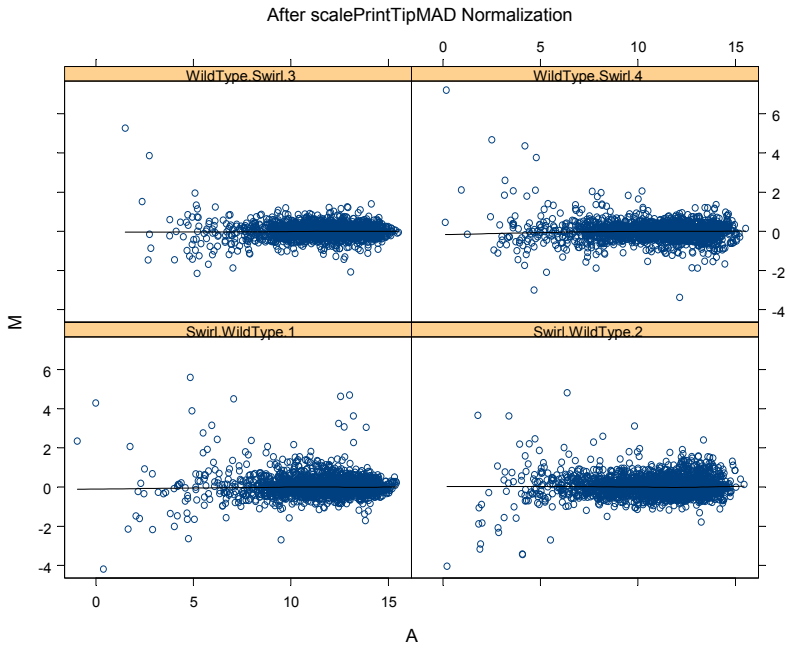


Figure 4.45: *Post-normalized M vs A plot for the swirl data.*

We can also do boxplots as a function of print-tip groups as follows:

```
> par(mfrow = c(1,2))
> maBoxplot(swirl.raw[,3], main =“Pre-normalization”,
      srt=90)
> maBoxplot(swirl.norm[,3], main =“Post-normalization”,
      srt=90)
```

The resulting graph is display is shown in Figure 4.46.

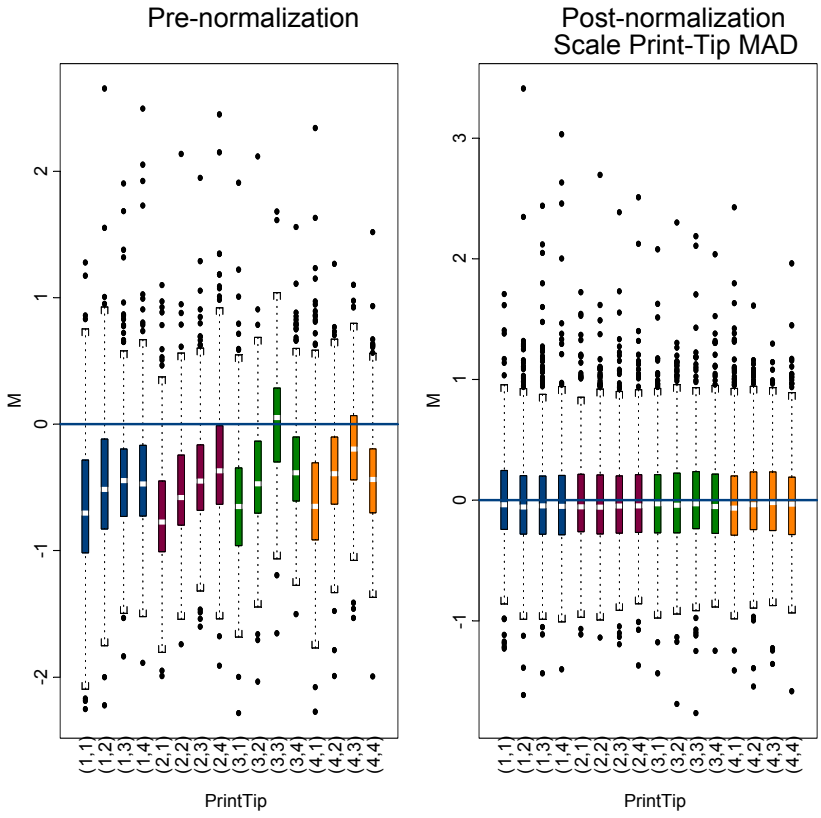


Figure 4.46: *Before and after scale print-tip MAD normalization of the swirl data.*

Before we move onto differential expression testing, note that the slots of the marrayRaw object are:

```
> getSlots(swirl.raw)
     maRf      maGf      maRb      maGb       maW
"matrix" "matrix" "matrix" "matrix" "matrix"

       maLayout      maGnames     maTargets      maNotes
"marrayLayout" "marrayInfo" "marrayInfo" "character"
```

Each of the first four slots are raw intensity matrices with dimensions equal to number of genes x number of chips after controls have been removed. For this example 7680 x 4.

```
> dim(swirl.raw@maRf)
[1] 7680    4
```

Once we apply the normalization procedures, background correction is done, the raw intensities are converted to M and A values, and the and the normalized object has different slot names.

```
> getSlots(swirl.norm)
      maA       maM    maMloc maMscale       maW       maLayout
 "matrix"  "matrix"  "matrix"  "matrix"  "matrix""marrayLayout"

     maGnames     maTargets      maNotes maNormCall
 "marrayInfo" "marrayInfo" "character"     "call"
```

Differential Expression Testing

Given that we have a *dye-swapped* cDNA experiment we'll use a traditional paired t-test to test for differences in expression. We first have to create a couple of objects that are arguments to the `aa.teststat` function.

Paired t-test

First extract M, the log intensity ratios.

```
### Extract M's = log2(R/G) for each chip
M <- maM(swirl.norm)
```

Now compute fold change in preparation to doing a paired t-test. See code for aa.teststat for details on how this is done in general.

```
### Compute fold change - prep to a paired t-test.
foldChange <- rowMeans(M[,c(1,3)] - M[,c(2,4)])
```

Get gene names and label the rows of the M matrix

```
### extract gene names
sw.probes <- maLabels(maGnames(swirl.norm))
dimnames(M)[[1]] <- sw.probes
```

Set up a factor which indicates which cell type is colored red.

```
### Set the factors - factor indicates which cell type is
colored Red
gfac <- factor(c("swirl", "wild-type", "swirl", "wild-
type"))

### Compute test statistics
testStat <- aa.teststat(M, gfac, test = "pairt")
```

Adjusted p-values Compute adjusted p-values for both Bonferroni and Benjamini and Hochberg methods.

```
### Compute adjusted p-values
rawp <- testStat[, "pValue"]
testObj <- mt.rawp2adjp(rawp, proc = c("Bonferroni","BH"))
```

We can now print the top 10 genes

```
> testObj@adjp[1:10,]
              rawp Bonferroni        BH
 [1,] 0.0002952891          1 0.6673955
 [2,] 0.0004345541          1 0.6673955
 [3,] 0.0005307666          1 0.6673955
 [4,] 0.0005736545          1 0.6673955
 [5,] 0.0006389848          1 0.6673955
 [6,] 0.0007114647          1 0.6673955
 [7,] 0.0007657569          1 0.6673955
 [8,] 0.0009233798          1 0.6673955
 [9,] 0.0009371951          1 0.6673955
[10,] 0.0010859229          1 0.6673955
```

Graphics We create several objects that get passed to the plotting functions

```
adjp <- testObj$adjp
index <- testObj$index

### Set up for Bonferroni adjustment plot
testDF<- data.frame(GeneName= sw.probes,testStat=
testStat[, "testStat"], foldChange= foldChange,
stringsAsFactors= F)
row.names(testDF) <- sw.probes
testDF<- testDF[index, ]
testDF[,"rawp"]<- rawp[index]
testDF[,"AdjPvalue"]<- adjp[, "Bonferroni"]

### Setup for BH adjustment plot
testDFBH<- data.frame(gName= sw.probes,testStat= testStat[,
"testStat"], foldChange= foldChange, stringsAsFactors= F)
row.names(testDFBH) <- sw.probes
testDFBH <- testDFBH[index, ]
testDFBH[,"rawp"]<- rawp[index]
testDFBH[,"AdjPvalue"]<- adjp[, "BH"]
```

Volcano plots

```
volcanoPlot(testDF, fwer = .5)
volcanoPlot(testDFBH, fwer = .5)
```

Graphlets

Now the Graphlets. The `diffExpr` object contain raw p-values and adjusted p-values for each procedure.

```
diffExpr <- data.frame(adjp)
row.names(diffExpr) <- sw.probes[index]

### Bonferroni adjustment
MultSumm <- multtest.graphlet(M, diffExpr[,2],
diffExpr[,1], index,
      testStat[,1], foldChange, fwer=0.05,
      procedure="Bonferroni", chip.name=NULL,
      volcano.plot=T, heatmap.plot=T,
      chromosome.plot = F, html.output=F,
      summary.name="MultSumm", open.browser=F)

### BH adjustment
MultSumm <- multtest.graphlet(M, diffExpr[,3],
diffExpr[,1], index,
      testStat[,1], foldChange, fwer=1,
      procedure="BH", chip.name=NULL,
      volcano.plot=T, heatmap.plot=T,
      chromosome.plot=F, html.output=F,
      summary.name="MultSumm", open.browser=F)
```

QUALITY CONTROL DIAGNOSTICS AND FILTERING

5

Quality Control Diagnostics **192**
Diagnostic Methods 192
Using the GUI 200

Filtering **203**
Array Filtering 203
Gene Filtering 204

QUALITY CONTROL DIAGNOSTICS

S+ARRAYANALYZER provides an assortment of graphical tools for assessing the quality of your experimental data. The tools allow you to consider quality in several ways so you can make judgements on the usefulness of the data in subsequent analyses and statistical inferences. You can examine an entire array as well as the individual genes or spots on an array. These tools range from color images of expression intensities where each pixel represents a different gene or spot on the array to sophisticated RNA degradation plots (for Affymetrix CEL data) and principal components plots.

In this section we describe the quality diagnostics tools available and how they can be used to assess the quality of your microarray data.

Once quality assessments are complete you can apply filtering methods to remove unwanted arrays and genes from your analyses. These methods are described subsequently in section Filtering on page 203.

Diagnostic Methods

The diagnostic methods vary slightly depending on the type of microarray data you have. For Affymetrix MAS data, the following methods are available:

1. Color image plot of the entire array.
2. M vs. A plot as either a scatter plot or a hexbin plot.
3. Genes Present plot.
4. Intensity boxplot.
5. RNA degradation plot.
6. Principal components plot.

Color Image Plot

The *color image* plot provides a *whole-array* view of expression intensities. Each pixel of the image represents the expression intensity of one spot on the array. The intensities are color coded to provide a quick visual inspection of the entire array. The pixels are arranged geographically just as they are on the array. Figure 5.1 shows an example image plot for Affymetrix CEL data.

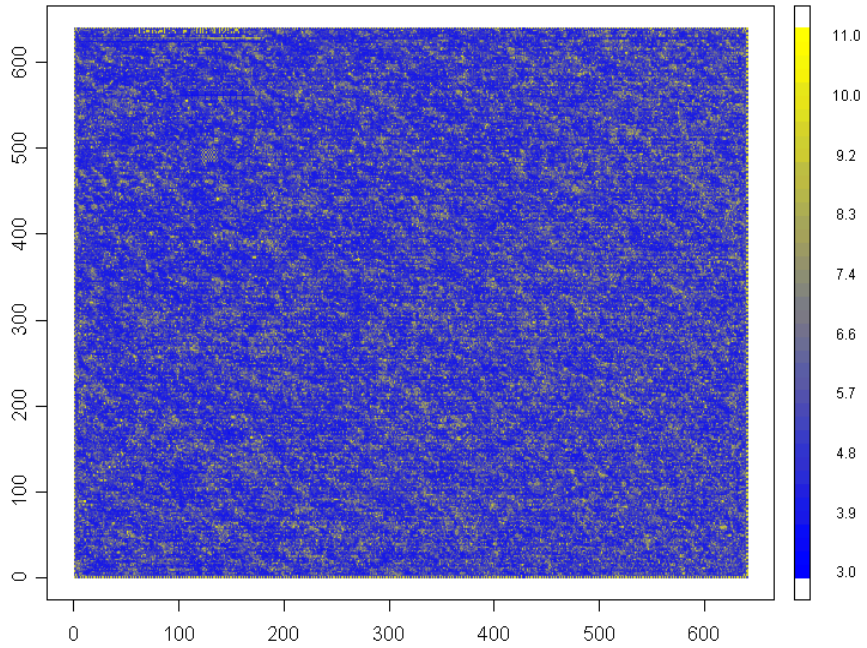


Figure 5.1: *Image plot example for Affymetrix CEL data.*

The image plot is ideal for discovering imperfections on the array surface such as scratches, contamination from debris and uneven hybridization. This is a useful plot for both Affymetrix and two-channel microarray data.

For two-channel data, with multiple print-tip groups, the image plot also shows the print-tip grouping. Figure 5.2 displays an image plot for a two-channel custom cDNA microarray with 16 print-tip groups.

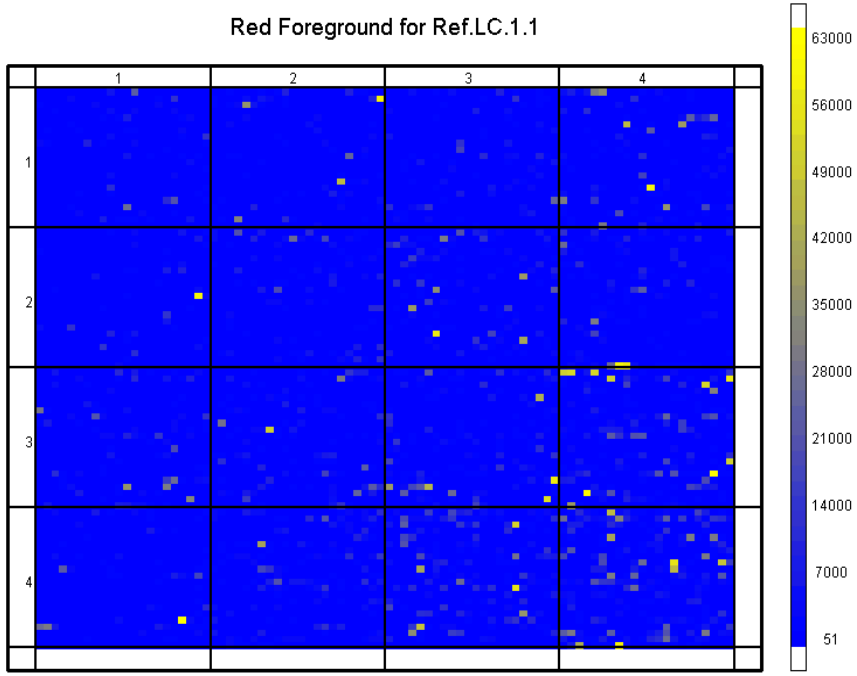


Figure 5.2: *Image plot example for two-channel data with 16 print-tip groups.*

M vs. A Plot

The MvA (or M vs. A) plot is a *scatter plot* of M versus A. For Affymetrix data, M and A are defined as:

$$M = \log_2(E_{ki}/E_{kj})$$

$$A = \frac{\log_2(E_{ki}E_{kj})}{2}$$

where E_{ki} is the expression intensity for replication i, $i \neq j$; i, j =1,..., # of replications of the k-th treatment.

For two-channel data, M and A are defined as:

$$M = \log_2\left(\frac{R}{G}\right) \text{ and } A = \log_2\sqrt{RG},$$

where R and G represent the expression intensities of the Red and Green channels.

The MvA plot is a plot of the fold change versus average $\log_2$ intensity for each pair of replications within each experimental condition. Figure 5.3 displays a typical MvAplot.

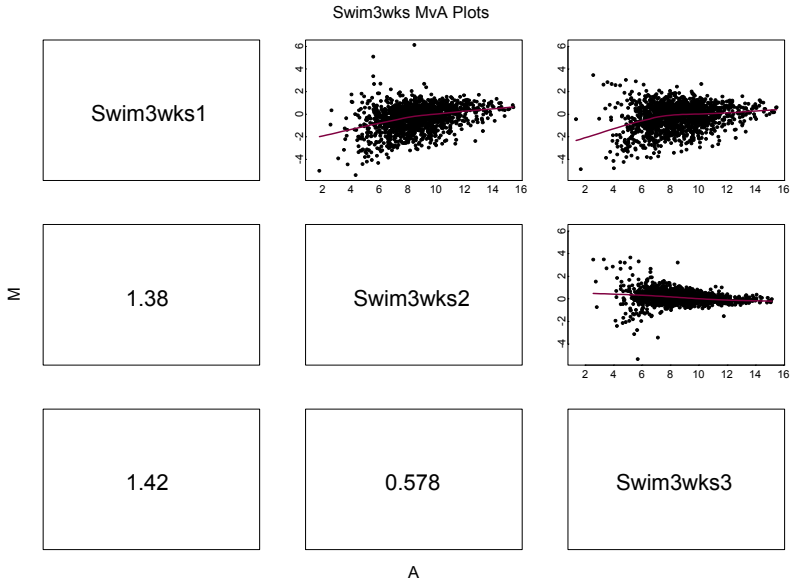


Figure 5.3: *An MvA plot for Affy MAS data.*

The points of MvA plots should hover around zero because no differential expression is expected for experimental replicates. The line going through the center of the points estimates the relationship between fold change and average log intensity between a pair of replicates. Deviations from zero suggest systematic bias indicating the need for normalization or possible removal of an array from subsequent analysis.

When MvA plots are plotted as *scatter plots* sampling is done to keep the plotting overhead to a reasonable level. The default sample percentage is 20%. For an array with 10,000 genes, 2,000 points are

plotted by default. The percentage of points plotted, referred to as the *sample size* in the GUI, is editable by the end user. Increasing the sample size to 100% is possible, but at the expense of increased overhead with little to no increase in useful information.

An alternative to the MvA scatter plot is the MvA hexbin plot which shows plotted "points" as colored hexagonal bins. The hexagonal bins represent the number of points at each bin location by the intensity and color of shading of the bin. A legend displays a set of colored bins and how they correspond to the density of points in each bin.

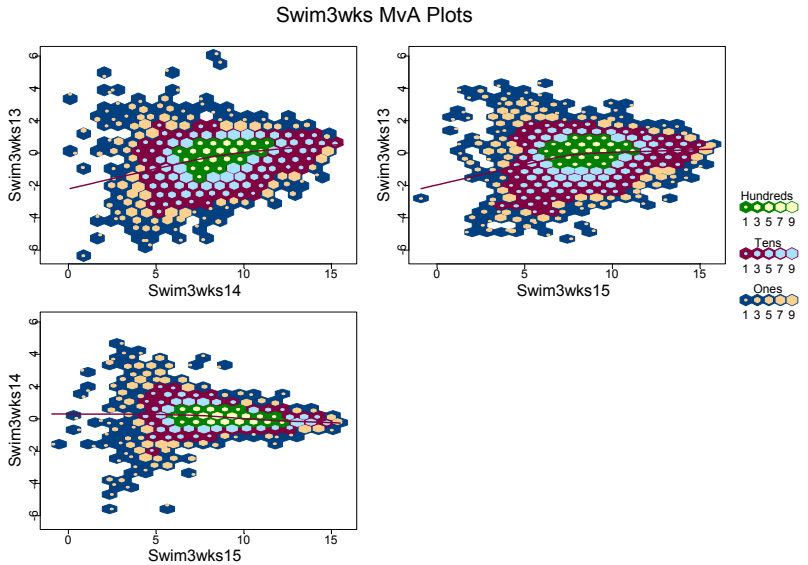


Figure 5.4: *An MvA hexbin plot.*

Genes Present Plot

The *Genes Present* plot is available for Affymetrix MAS data only if you have selected the **Detection** variable during import. See the section Keeping Extra Variables on page 30 in Chapter 2, Examples: Affymetrix MAS Data for more details on keeping extra variables.

The Genes Present plot displays the percent of genes detected on each array and gives a sense of the quality of the hybridization across the entire array. Figure 5.5 displays an example Genes Present plot.

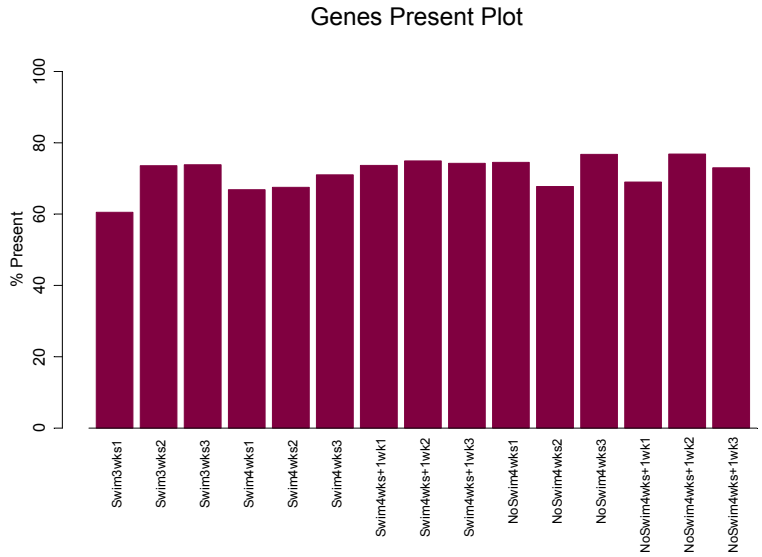


Figure 5.5: *An example Genes Present plot displays a bar plot of the percent of genes present on each array.*

Intensity Boxplot Box and whisker plots display distribution summaries for a set of numbers. The *Intensity Boxplot* displays a set of box and whisker plots, one for each array in the experiment so you can compare distributions.

A box and whisker plot displays the median (the dot or band in the center of the box), the25th and 75th percentiles (the lower and upper shoulders of the box) and the "extremes" (minimum and maximum unless outliers are present). When outliers are present the whiskers are drawn at the nearest value not beyond 1.5 * IQR from the quartiles, where IQR is the inter-quartile range, i.e., the difference between the 75th and 25th percentiles. Figure 5.6 displays an example Intensity Boxplot.

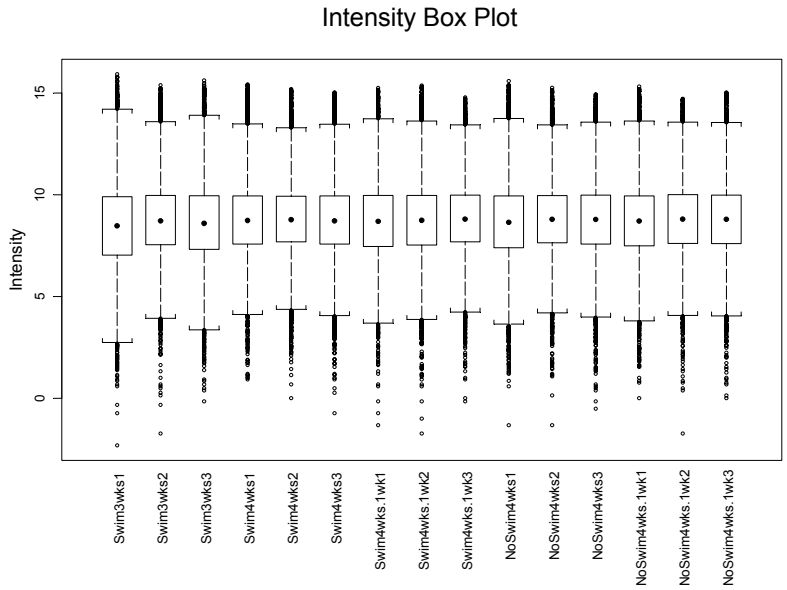


Figure 5.6: *Example Intensity Boxplot for comparing distributions of expression intensities.*

RNA Degradation Plot

The *RNA Degradation* plot is only available for Affymetrix probe-level (CEL) data. The RNA Degradation plot displays the average expression intensity for a set of marker probes at each location of the probe set. Separate profiles (means at probe locations connected by lines) are plotted for each array. Trends in the profile lines, such as those displayed in Figure 5.7 indicate differential labeling as a function of location in the probe set. If there is no differential labeling you should expect to see relatively flat profiles.

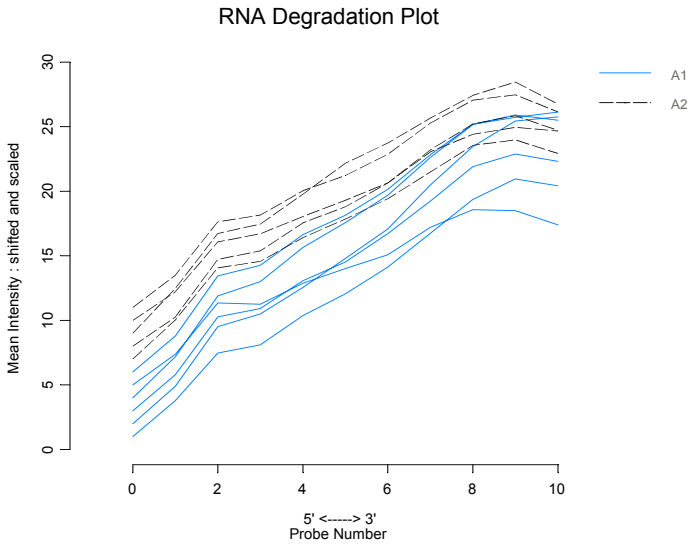


Figure 5.7: *Example RNA Degradation plot indicating differential labeling as a function of location in the probe set.*

Prin. Comp. Plot

The *Principal Components* plot is a plot of the eigen vectors resulting from a principal components analysis taking each chip as a variable with values equal to the expression intensities for all genes present on the array. The plot uses different plotting symbols for each experimental condition so association between treatment conditions is indicated by relative position in the graph. Replicates with wildly deviant points indicate potential problems with one or more arrays.

The example shown in Figure 5.8 shows very different expression for arrays taken later in the study (27 and 31) compared to arrays taken earlier in the study (1, 7 and 11).

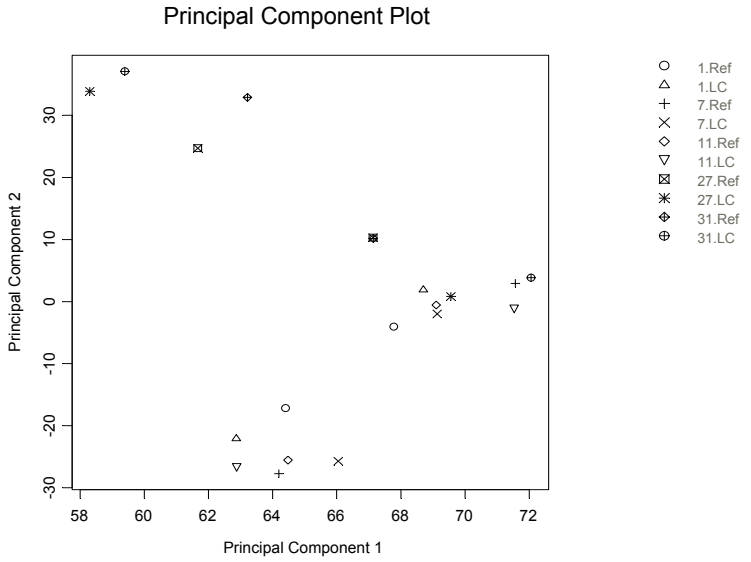


Figure 5.8: *Example Principal Components Plot.*

Using the GUI

The **Quality Control Diagnostics** dialog is available by clicking **ArrayAnalyzer ▶ QC Diagnostics ▶ <Affymetrix, Two Channel>** from the main S-PLUS menu bar. The use of < > indicates that one of **Affymetrix** or **Two Channel** is selected but not both.

Diagnostics for Affymetrix Data

Figure 5.9 shows the selection of the Affymetrix **Quality Control Diagnostics** dialog.

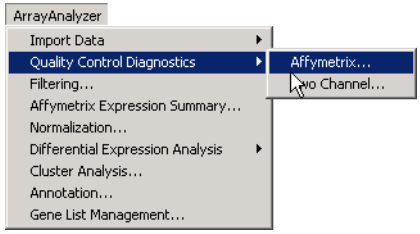


Figure 5.9: *Selecting **Quality Control Diagnostics** for Affymetrix data.*

The resulting dialog has four groups of controls:

1. Data

2. Image plot
3. MvA plot (Bland-Altman)
4. Other plots

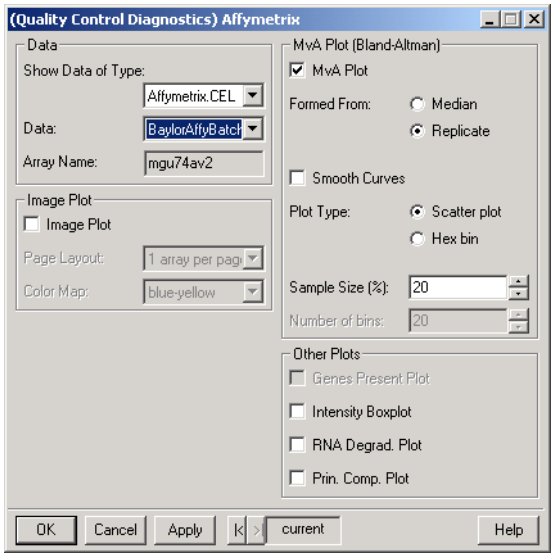


Figure 5.10: *The **(Quality Control Diagnostics) Affymetrix** dialog.*

Selecting Data

Select data by choosing **Affymetrix Summary** or **Affymetrix CEL** from the **Show Data of Type** drop-down list and then select the data of your choice.

Selecting Graphical Options

Now choose the plots you wish to display. Note the page layout options for the image Plot - one or four images per page. Also, note that for the MvA plot you can produce either a scatter plot, with the degree of sampling specified in the **Sample Size** field, or a hexbin plot with the number of bins specified by the **Number of bins** field.

Clicking **Apply** or **OK** generates the plots you selected. Be aware that RNA Degradation and Principal Components plots may take some time to complete.

Diagnostics for Two-Channel Data

The use of the **Quality Control Diagnostics** dialog for two-channel data is identical to that for Affymetrix. However, there are a couple of options missing. There is not a Genes Present plot nor an RNA Degradation plot. Figure 5.11 displays the dialog with some options selected.

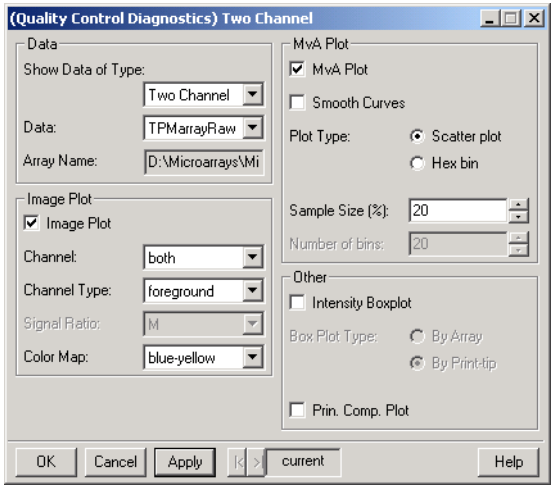


Figure 5.11: *Example dialog for* ***Quality Control Diagnostics*** *for two-channel data.*

FILTERING

S+ARRAYANALYZER provides simple mechanisms for removing arrays or genes or both before proceeding with an analysis. Because of the way S-PLUS saves data on hard disk, you don't have to worry about losing your original data. The default object name for the result of any filtering operation is different than the original data object so the original data stays intact as long as you preserve the original name.

To start a filtering operation, click **ArrayAnalyzer ► Filtering** from the main S-PLUS menu. The resulting dialog has two pages: One for samples or entire arrays, and one for genes.

Array Filtering

Filtering on arrays or samples is straight-forward:

1. Select the type of data from the **Show Data of Type** drop-down list. Note that only **Affymetrix Summary** and **Two-Channel** data are selectable.

Note
If you know (based on QC Diagnostics or by other means), that you want to delete an array from a probe-level data set, you can do that once it is summarized.

2. Select the data object you want to filter from the **Data** drop-down list.
3. Select the samples you want to drop from the analysis by selecting them in the **Samples to Keep** list and moving them to the **Samples to Drop** list by clicking on the **Drop** button.

Note
Once you've selected a sample and moved it into the **Samples To Drop** list, you can move it back into the **Samples to Keep** list by selecting it in the **Samples to Drop** list and clicking the **Keep** button. Note the sample appears at the bottom of the list, rather than in the original order.

Figure 5.12 displays the **Filtering** dialog set up to drop three arrays before running an analysis.

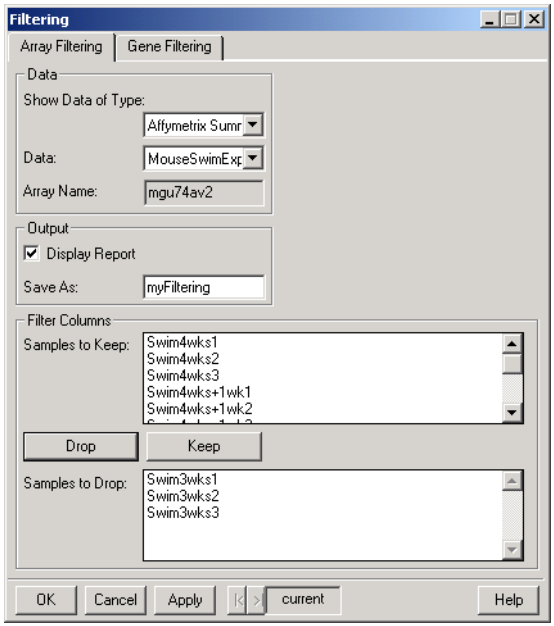


Figure 5.12: ***Filtering*** *dialog set up to remove three arrays before running analyses.*

Gene Filtering

To filter on genes, you create an S-PLUS logical expression that specifies the condition or conditions for keeping or deleting genes. Some examples include the following:

```
Expr.Intensity > 2
RedForegnd > 5000 & M > 2
Flags = = 0 InAtLeast 1
```

You can type expressions directly into the **Expression** field on the **Gene Filtering** tab or build it sequentially. In the following example, we run a filtering experiment from Chapter 4, Examples: Two-Color Data. Refer to the chapter to create the data set, **TPMarrayRaw**, and follow the instructions below for creating the filtering expression.

For this example, use the **Flags** column to create expressions for eliminating rows with non-zero flags:

```
Flags = = 0 InAtLeast 1
```

Here are the steps to create it:

1. Select a data column, **Flags**, and click the **Add** button in the Data group to add the name **Flags** to the **Expression** field at the bottom of the dialog.
2. Select the '= =" symbol (double equal sign) from the **Logical** field of the **Operations** group and click the **Add** button right under **Logical** label to add the = = symbol to the **Expression** field at the bottom of the dialog.
3. Select the value 0 (zero) from the list in the **Column Values** group and click the **Add** button in that group to add 0 (zero) to the **Expression** field at the bottom of the dialog.
4. Select 1 (one) in the **InAtLeast Value** field of the **Operations** group and click the **Add** button right under the **InAtLeast Value** label to add **InAtLeast 1** to the **Expression** field at the bottom of the dialog.
5. Ensure that the **Values Selected** in the **Expression** group is set to **Keep**.
6. Click **OK** to eliminate all genes except those with `Flags = = 0`.

Note

The **InAtLeast 1** operation allows you to keep any gene that has a **Flags** value of zero on one or more arrays. Without this operation, a gene is eliminated unless the **Flags** value is zero on all arrays.

This operation works similarly but in reverse when you are dropping genes rather than keeping them. When dropping genes, without the **InAtLeast1** operation genes will only be dropped when the value is equal to the zero (for this case) in all samples.

The resulting **Gene Filtering** tab of the **Filtering** dialog is displayed in Figure 5.13.

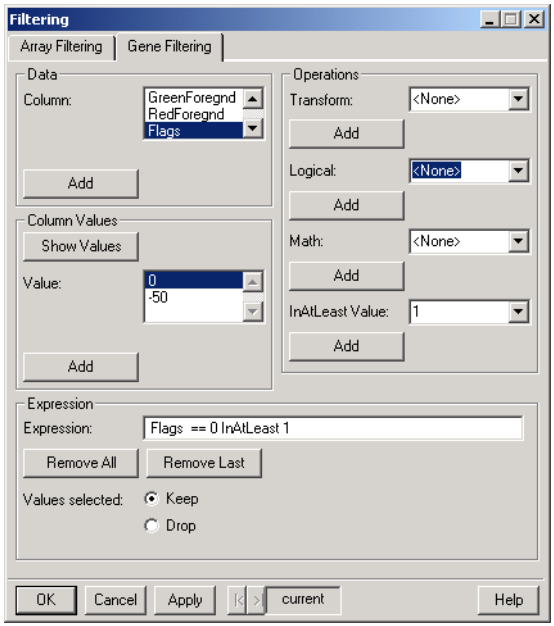


Figure 5.13: *Filtering genes with non-zero flags.*

PRE-PROCESSING AND NORMALIZATION

6

Introduction **209**

Normalization **210**
- Technical Sources of Variability 210
- Why do We Normalize Data? 211
- Normalizing in S+ArrayAnalyzer 212

Ideas in Normalization **214**
- Normalizing to One Point 214
- Normalizing to Many Points 215
- Cautions in Normalizing 216
- Workflow 217

Diagnostic Plots **218**
- Box Plots 218
- MvA Scatter Plots 218
- MvA Hexbin Plots 219
- Chip Specific Plots 219

Normalization Methods for Two-Channel Data **220**
- Normalizing with the GUI 220
- Notes For Command Line Users 221
- Two-Channel Diagnostic Plots 221
- Location and Scale Normalization (Within Array Normalization) 225
- `maNormMain` Function 226
- Normalization Functions: `maNorm`, `maNormScale` 228
- Normalization Between Arrays 230

Pre-Processing and Normalization for Affymetrix Probe-Level Data **232**
- CDF and Probe Libraries 233
- Affymetrix Diagnostic plots 234

Background Correction 237
PM Correct Methods 239
Normalization 241
Summarization Methods 245

Normalization Methods for Affymetrix MAS Data 251
Normalization Methods 252
Diagnostic Plots for Summarized Affymetrix Data 254

References 256

INTRODUCTION

Before differential expression testing can be performed, microarray data must be pre-processed and normalized. Pre-processing refers to the process of correcting the measured spot intensities for background signal and non-specific binding and, for probe-level data, summarizing the multiply cloned gene expression measurements into one expression measure. As described in Parmigiani et al (2003), data from microarrays are subject to many sources of extraneous variability including manufacturing; preparation of mRNA from experimental samples; hybridization; scanning; and imaging. These sources of variability are often called *technical sources of variability*. The removal and balancing of extraneous, technical variability before analysis allows for more confident interpretation of the estimated differential expression effects as true differential expression and not a result of systematic experimental artifacts.

Pre-processing of probe-level data primarily involves summarizing data from probe sets[1] into a single measure per gene/transcript. The Affymetrix MAS software provides a way of doing this based on a one-step Tukey biweight procedure. Other approaches including the MBEI method of Li and Wong (2001) and the RMA method of Irizarry et al. (2003b), have been shown to provide improved extraction of biological information from probe-level data (Irizarray et al., 2002, 2003b).

This chapter address the variety of methods available in S+ARRAYANALYZER for correcting, normalizing and summarizing (probe-level) microarray data. The S-PLUS script **splus62/module/ArrayAnalyzer/examples/scripts/normalization_chapter.ssc** includes the example code presented in this chapter. Some of the example computations are time-consuming.

1. Affymetrix GeneChip® microarrays represent each gene with an oligonucleotide (25-mer) probe spotted at typically 16-20 pairs of spots (32-40 spots in all). Each probe pair consists of a spot for the probe, called a *perfect match* (PM) and a spot for a slight alteration of the probe, called a *mismatch* (MM). The collection of the PM and MM spots for a specific gene are called a *probe set*.

NORMALIZATION

Many factors can modify spot intensity other than differential gene expression. And each type of microarray chip has its own inherent systematic variations that need to be taken into account. Normalization can be thought of as a series of corrections for these systematic effects. In general, normalization is needed to ensure that observed differences in fluorescence intensities are due to differential expression and not experimental artifacts, and should be done before any analysis that compares gene expression levels within or between arrays.

Effects that have been consistently removed by normalization include differential (nonlinear) hybridization of the two channels in two-color cDNA arrays; biases in DNA spotting due to eroded print tips; and spatial variability of signal in regions of an array.

Technical Sources of Variability

Two-Channel Data

Two-channel microarrays developed within research organizations are subject to variability in all of the preparation phases e.g. amplification, purification and concentration of DNA clones; the amount of DNA spotted, the binding of the DNA to the array, the shape/size of the spot and dye quality and labeling. There are several environmental factors at play during hybridization and scanning including temperature, humidity, non-specific binding and washing conditions. The scanning process is complex, with higher intensities giving higher signals but leading to saturation at the high end, while lower intensities remove saturation but miss signal on the low end. Imaging algorithms are likewise complex, with significant segmentation issues involved in the separation of signal from background (see Yang et al., 2001).

Affymetrix Data

Commercially manufactured oligonucleotide arrays have their own variability issues including those described above. Affymetrix, the market leader, has made a considerable effort to minimize and track variability in their arrays. As in many assay formats, the vendors of

microarray technology compete based on how well their manufacturing and deployment processes control extraneous variability and provide reproducible results.

Why do We Normalize Data?

Let's look at the swirl data set that has already been discussed in section Mutant Zebra Fish Data on page 127. Figure 6.1 and Figure 6.2 shows a box plot for each chip in the experiment and an M vs. A plot of one of the chips. Figure 6.1 shows that there are significant differences in the median log-intensity differences. If the probes are placed randomly on the slide, and the experimental conditions are well controlled we would expect the medians of each print-tip group to be similar. However, the experimental conditions are not perfectly controlled, as shown by the negative values for all of the print-tip groups in Figure 6.1. The log-ratio of intensities is a measure of the difference between the red and green fluorescence. The fact that this quantity is always negative suggests an imbalance in intensities of the two dyes, Cy3 and Cy5.

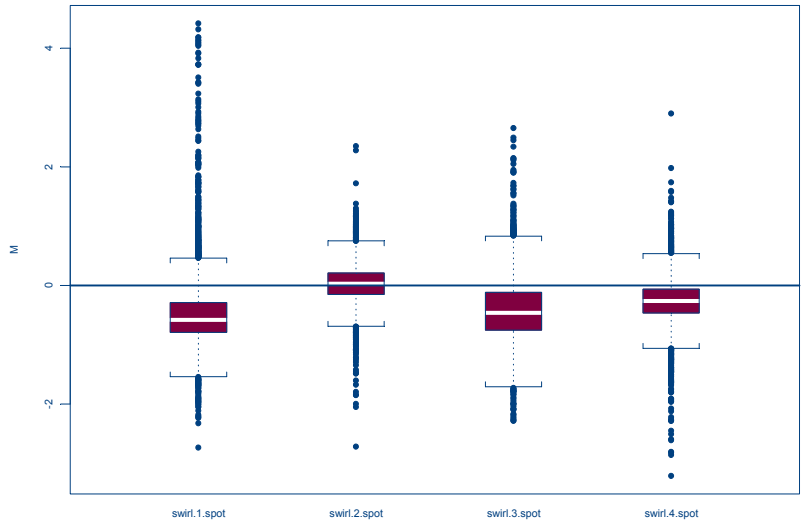


Figure 6.1: *Box plot for swirl experiment before normalization*

Figure 6.2 is an M vs. A plot for one chip in the swirl set. The loess curves for each print-tip are superimposed on the scatter plot. The plot shows a non-linear dependence of the log-ratio of red to green intensity (M) on the average log intensity (A). In section Normalization Methods for Two-Channel Data on page 220 we examine a number of normalization methods for this dataset to correct this systematic variation.

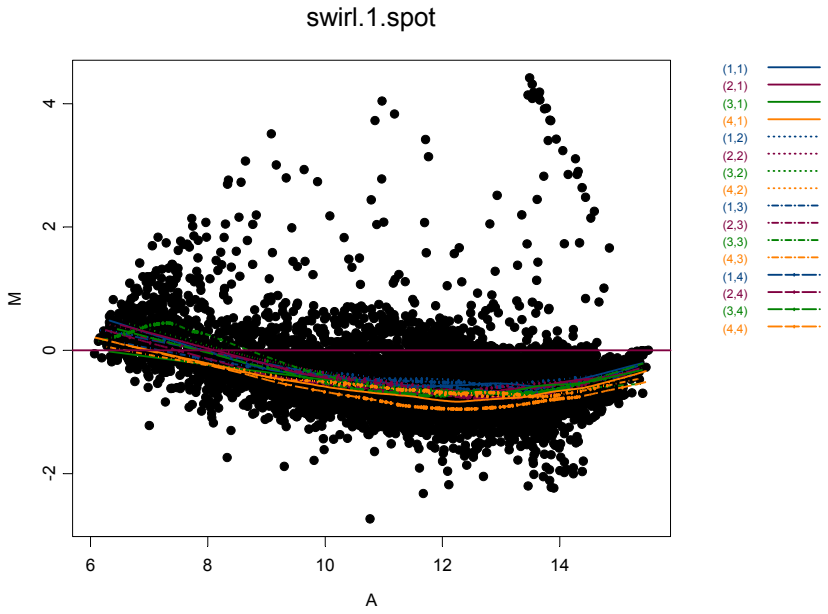


Figure 6.2: *MvA plot for chip 81 of swirl dataset before normalization with loess curves for each print-tip group.*

Normalizing in S+ArrayAnalyzer

S+ARRAYANALYZER provides a wide variety of normalization methods depending on the form of the data prior to normalization. For two-channel chips the scanning software typically accounts for background noise and adjusts for control information. For Affymetrix probe-level data (.CEL files) the analyst needs to account for background noise and may make use of controls (often the mismatch probes, MM, are used for this) to correct for random, non-specific binding. The analyst must also summarize the probe-level data into a single value per gene/transcript. Affymetrix summarized data (e.g.,**.chp** files output from MAS 4/5 software) has already been

background adjusted and perhaps mildly normalized; and the probe-level data have been summarized into a single intensity value per gene/transcript.

Because of the inherent differences in two-channel data versus Affymetrix data, the specifics of the normalization methods differ between data types. However, normalization in general can be thought of as either normalization to a point (location normalization) or scaling of the variability of the data (scale normalization). Having a visual representation of the data is very useful in the normalization process. S+ARRAYANALYZER includes a variety of diagnostic plots and the sections that follow discuss the diagnostic plots and specific methods for normalizing two-channel and Affymetrix data.

IDEAS IN NORMALIZATION

Normalization typically involves adjusting distributional summaries of data from each chip to common reference values. Sometimes reference values are supplied by the user. Alternatively, some methods assume one chip is the reference chip and the other chips in the set are then normalized to the target chip's reference values.

A median is a robust estimate of the center of the data distribution where just under 50% of the data on either side of the median can be moved to infinity and the median value will not be affected. It is a quantity that defines the center of the data - 50% of the data are above the median and 50% are below the median. Consequently, the median is often used as a reference value. The inter-quartile range (IQR) estimates the spread or variability of the data and is computed as the range of the middle 50% of the data. The IQR is a robust estimator of spread in that you can move just under 25% of the data at either end of the distribution to infinity and the IQR remains unchanged. The robust properties of the median and IQR make them good reference values in normalization procedures.

There are also methods which do not require a target, or reference chip. These methods use the information from the chip set to be normalized to create average reference values for the chip set.

Normalizing to One Point

We can think of normalizing groups of data to a reference point - that is, bringing the median of a data group to a fixed reference point through a shift of the values. This reference point can be as simple as a given constant intensity value (constant or median normalization) or as complicated as fitting a locally-weighted least squares regression (loess normalization) through the data. We call this type of normalization *location normalization*. Location normalization is necessary to correct for spatial variation, e.g. such as when the slide is slightly tilted during hybridization, which results in more mRNA available for binding at different locations on the slide.

Normalization Using Loess

One of the most common location normalization methods for microarray data is to normalize the data to a loess curve fit through the MvA plot. The loess method fits a curve to the data using robust locally weighted regression as discussed in Cleveland (1979); Yang et al. (2001, 2002) and the *Guide to Statistics*. Local regression is a

smoothing method for summarizing multivariate data using general curves and surfaces. The smoothing is achieved by fitting a linear or quadratic function of the predictor variables locally to the response data (M in this case). The loess procedure fits polynomials over contiguous subsets (intervals in one dimension) of the predictor values using iteratively weighted least squares. Yang and Dudoit (2003) write that, in the context of microarray experiments, robust local regression allows us to capture the non-linear dependence of the intensity log-ratio, M, on the overall average intensity, A, while at the same time ensuring that computed normalization values are not driven to a small number of differentially expressed genes with extreme log-ratios.

Normalizing to Many Points

Expression data may be variable between chips not only in the median of the data, but also in its spread around that median value. Variability may be due to such things as scanner settings and different concentrations of mRNA across slides. In order to compare expression across slides, these extraneous effects must be minimized.

The spread of the data in some range can be scaled to be the same between groups by specifying that the data between groups match at more than one point. For example we could specify that the IQR of the data be the same. This requires that the data be scaled so that the spread of the middle 50% of the data is identical across the groups.

We can extend this idea of normalization so that the data matches at a sequence of points, not just one or two points. For example, deciles (10th, 20th, 30th, ..., 90th percentiles) may be aligned via this type of normalization.

Normalization Using Quantiles

The quantiles method in S+ARRAYANALYZER extends the many point approach described above to a fine-scale granularity such that each ordered individual gene expression value is aligned. The method assumes there is an underlying common distribution of intensities across all chips in the set, and disparate detests can be transformed to the same distribution by transforming the quantiles (at the level of individual values) of each to have the same value. Details of this transformation can be found in the `normalize.quantiles` help file references. The draw back of this method is that extreme values in the tails are normalized to the same values, thus possibly loosing the differential expression information. Empirical evidence, however, suggests that this is not a problem (see Bolstad et. al. (2002)).

Normalization using VSN

VSN is short for *variance stabilization normalization*. In the VSN normalization method, the intensities from each array are calibrated by a suitable affine transformation, then transformed by a variance stabilizing transformation. After this, systematic array- or dye-biases should be removed, and the variance should be approximately independent of the mean intensity. This is useful for subsequent analyses such as hypothesis tests, ANOVA modeling, clustering, or classification that assume that the variance is the same for all observations. Note that VSN only addresses the dependence of the variance on the mean intensity. There may be other factors influencing the variance, such as gene-inherent properties, or changes of the tightness of transcriptional control in different conditions. If necessary, these need to be addressed by other methods. For additional information on this method please refer to Huber (2002 and 2003).

VSN works on the raw data from two-channel, affymetrix summary and affymetrix probe-level sources. For Affymetrix summary data, VSN returns data on a generalized log scale. For Affymetrix probe-level data, VSN works like other probe-level normalization methods and returns data on an exponentiated generalized log scale - this allows for subsequent logging that occurs in probe-level summary methods. For additional information on this method please refer to Huber (2002 and 2003).

WARNING:

Because VSN works on raw data, be sure that when using VSN on Affymetrix summary data, the log transform of the data is not done during import.

From the command line VSN normalization can easily be done on `marrayRaw` class objects using the `vsn` function or by coercing an `marrayNorm` object to an `exprSet` object and passing that to `vsn` (See the `vsn` help file).

Cautions in Normalizing

Some care is required when normalizing across treatment groups to not wash out signal, particularly for aggressive normalization approaches such as the quantile method. This is not much of an issue with mild normalization approaches such as lining up medians and IQR's. In the S+ARRAYANALYZER GUI, normalization is performed

for the set of chips provided, i.e., all those read into one object during the data import phase. **Thus, if the whole experiment is supplied, normalization will be done across treatment groups.** From the command line, the user has more control. For example, the user may choose to normalize within experimental conditions and merge the resulting normalized data.

When chips are normalized to an average reference, it is assumed that there is a common underlying intensity distribution on each chip. For this reason, pairwise normalization, where one chip is a target chip, may be preferable when there are just 2 chips. But pairwise normalization when there are more than two chips has been shown to give variable results depending on which chip is chosen to be the reference chip (Bolstad, et. al., 2002).

Normalization of microarray data is currently an active research topic. We leave it to the researcher to decide the best approach for their data. *Examples shown in this chapter are for demonstration purposes only.*

Workflow

Data corrections and normalizations can be done in series. The suggested work flow is as follows:

For Affymetrix probe level data:

- Background correcting.
- Probe-level summary (Summarizing the 11-20 probe pair sets into a single value for each transcript).
- Location/scale normalization.

Affymetrix summary data (e.g., **.CHP** file data from Affymetrix 4/5); and two-channel summary data (**.GPR** file data from GenePix®) are typically normalized to location and possibly scale before differential expression analysis.

DIAGNOSTIC PLOTS

Diagnostic plots of intensity data can help identify printing, hybridization, scanning artifacts and other sources of unwanted variability which can removed before analysis of differential gene expression. The S+ARRAYANALYZER GUI provides a variety of diagnostic plots to help identify such unwanted variability and guide subsequent adjustments and modeling procedures. Please refer to Chapter 5, Quality Control Diagnostics and Filtering for additional details on these diagnostic plots. Additional plots such as Histograms are also available from the S+ARRAYANALYZER command line.

Box Plots

Box plots show side-by-side graphical summaries of intensity information from each array. The summary consists of the median and the upper and lower quartiles (75th and 25th percentiles, respectively) of the data. The central box in the plot represents the inter-quartile range (IQR), which is defined as the difference between the 75th percentile and the 25th percentile. The median is represented by a line or a dot in the middle of the box. By default, the upper and lower whiskers on the box plots are placed at the most extreme observation not exceeding plus and minus 1.5 times the IQR from the quartiles. Data outside the whiskers are plotted separately. The y-axis is typically the intensity log-ratios and the x-axis is the grouping variable. Figure 6.1 on page 211 shows an example of a typical box plot for an experiment with two replicate slides for each dye-swap condition.

MvA Scatter Plots

Scatter plots of spot statistics allow the user to highlight and annotate subsets of points on the plot, and assess patterns of differences in intensities between channels or chips. Such patterns may be visualized via fitted curves from robust local regression or other smoothing procedures. The MvA plot shows the log-ratio of the intensities (difference of the log intensities usually termed M) between channels or chips to the average of the log intensities (usually termed A) for the channels/chips. Figure 6.2 shows an MvA plot for one chip in the swirl dataset (two-channel data) with loess curves overlaid for each print-tip group.

Note

MvA plots from the GUI plot a maximum of 2000 genes. If there are more than 2000 genes in the experiment, then only 2000 randomly sampled genes are plotted.

MvA Hexbin Plots

Hexbin plots provide another dimension to plotting data. Unlike the MvA scatter plots which use no more than 2000 points, hexbin plots use ALL the data and show its density or concentration in a particular area by coloring that area. So hexbin plots not only let you see ALL the data but they let you see its concentration. Hexbin MvA are important to show a global picture of the data before and after normalization.

Chip Specific Plots

Detailed information about these plots for the different chips types are available in the following sections. The sections that follow will also describe other types of plots that are available from the command line.

NORMALIZATION METHODS FOR TWO-CHANNEL DATA

There are often systemic variation and imbalances of the red and green fluorescence intensities in two-channel data. This variation is usually not constant across the spots within or between arrays, and can vary according to overall spot intensity, location on the array, plate origin, and possibly other variables. Some causes of the imbalances may be the following:

- Labeling efficiencies and scanning properties of the Cy3 and Cy5 dyes.
- Amounts of Cy3- and Cy5-labeled mRNA.
- Scanning parameters, such as PMT settings.
- Print-tip, spatial, and plate effects.

Normalizing with the GUI

The GUI performs default setting normalization for a batch of arrays. For two-channel arrays the chips need to first be normalized to balance differences in the red and green channels on each chip. After that, if desired, the set of arrays can be normalized to clean up undesirable, between array variations. The GUI includes the within array methods listed in Table 6.3 and Table 6.4. The GUI also allows for between array normalization via quantiles after within normalization is completed. Normalization between arrays is needed if downstream analyses (e.g. ANOVA) compare experimental conditions that vary between arrays. These options are obtained by selecting a method for the **Between Array** field shown in Figure 6.3.

Please refer to section Normalization on page 210 for examples showing how to use the GUI to produce diagnostic plots and normalize two-channel data.

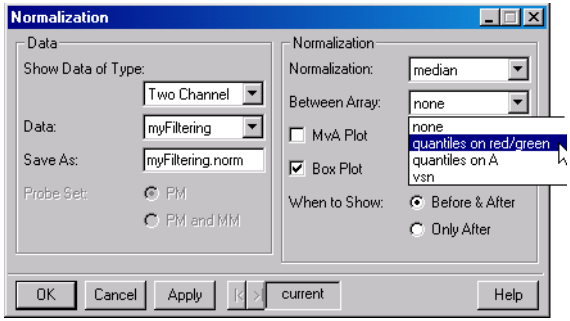


Figure 6.3: ***Normalization*** *dialog for two-channel data where an optional between array method is chosen in addition to within array normalization.*

Notes For Command Line Users

The normalization and plotting functions for two-channel data make heavy use of the accessor methods for the different `marray` classes. The input parameters to the functions are labeled `x`, *y* and `z`. Each function uses the `x`, *y* and `z` parameters differently, refer to the help files for specifics. In general, these parameters give the accessor methods for the `marrayRaw` or `marrayNorm` class objects. These accessor methods are then used to extract the desired information from the data object for use in the normalization computation or plotting function. Useful methods are: `maM` and `maA` for obtaining the intensity log-ratios and average log-intensities respectively and the `maPrintTip` method, which computes the print-tip grid coordinates for the spots. `maPrintTip` and `maPlate` are used to stratify the data by print-tip groups or by chips. Please refer to the `marrayClasses` documentation (**splus62/library/marrayClasses/marrayClasses.pdf**) or the `marrayRaw` and `marrayNorm` help files for additional options.

Two-Channel Diagnostic Plots

Normalization usually begins with exploratory data analysis and diagnostic plots. Two-channel data typically includes two treatment conditions on one chip. Dudoit et al (2002) and Yang et al. (2001) suggest that the most useful way to view such data in order to identify spot artifacts, and for normalization purposes, is via an M vs. A plot of

the intensity log-ratio $M = \log_2\left(\frac{R}{G}\right)$ vs. the mean log-intensity $A = \log_2\sqrt{RG}$. This amounts to a 45 degree counter-clockwise rotation of the ($\log_2$G, $\log_2$R)-coordinate system, followed by a scaling of the coordinates. This plot highlights the difference between the red and green channels as a function of average intensity across the two channels. Figure 6.2 on page 212 shows an example of an MvA plot for two-channel data.

Box plots are also available from the normalization dialog. The y-axis typically shows the intensity log-ratio (M). The x-axis shows the grouping (chip or print-tip). Please see section Box Plots on page 218 for additional information on box plots. From the command line we can create a box plot by print-tip groups for chip 93 of the swirl dataset discussed in section Mutant Zebra Fish Data on page 127 as follows (Information on the swirl data set is also available in the swirl help file, type `help(swirl)`)

```
> maBoxplot(swirl[,3], main ="Swirl array 93:  pre-
normalized")
```

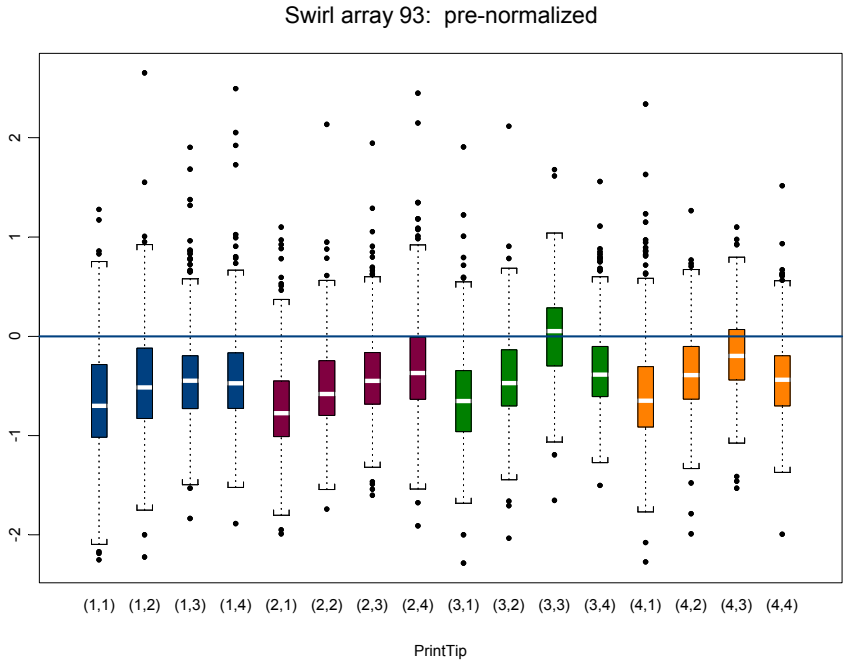


Figure 6.4: *Box plot for chip 93 of swirl dataset before it is normalized.*

Plots for two-channel data are available from the command line using the functions in Table 6.1. Following are examples of how to use these functions. Please refer to the marrayPlots library pdf (**splus62/library/marrayPlots/marrayPlots.pdf**) and the marrayPlots library help files for detailed descriptions of the function arguments. The section Notes For Command Line Users on page 221 discusses the meaning of the `x`, `y` and `z` parameters used in the plotting functions.

Table 6.1: *Plotting functions available for two-channel data.*

Plotting Function	Default Parameter Settings	Description
`maPlot`	`x="maA", y="maM", z="maPrintTip"`	Produces scatter-plots of microarray spot statistics for the classes marrayRaw, marrayNorm and marrayTwo. Creates plot for first chip given.
`maBoxplot`	`x="maPrintTip", y="maM"`	Produces box plots of microarray spot statistics for the classes marrayRaw, marrayNorm and marrayTwo. Plots by print-tip groups if given one chip. Creates one box plot for each chip if given more than one.
`maImage`	`x="maM", subset=TRUE, col, contours=FALSE, bar=TRUE`	Creates spatial images of shades of gray or colors that correspond to the values of a statistic for each spot on the array. The statistic can be the intensity log-ratio M, a spot quality measure (e.g. spot size or shape), or a test statistic. This function can be used to explore whether there are any spatial effects in the data, for example, print-tip or cover-slip effects. Creates plot for first chip given.
`maDotPlots`	`data, x = list("maA"), id = "ID", pch, col, nrep = 3`	A dot plot showing the values of replicated control genes

```
# Additional examples of diagnostic plots
#
# Boxplots of all chips in swirl dataset
# - pre-normalized swirl dataset
> maBoxplot(swirl)
#
# Boxplots of pre-normalization red foreground intensities
# for each grid row for the Swirl 81 array.
> maBoxplot(swirl[,1], x="maGridRow", y = "maRf",
  main = "Swirl array 81: pre-normalization red
  foreground intensity")

# MvA plot of chip 93, overlaid with
#    loess curves for each print-tip group (default)
> maPlot(swirl[,3])

# Image plots of chip 81, the first chip in the object
> maImage(swirl, x="maGf") # green foreground plot
> maImage(swirl, x="maM")  # log-intensity ratio plot
```

Location and Scale Normalization (Within Array Normalization)

Loess Normalization

S+ARRAYANALYZER supports both location and scale normalization for two-channel data. These normalization methods help normalize the red and green channels on one chip (within chip normalization). One of the most common location normalization methods is loess normalization. This method normalizes the data to the loess curve on the MvA plot. (Please refer to the section Normalization Using Loess on page 214 for more details about loess curves.) For two-channel data the intensity log-ratio $M = \log_2\left(\frac{R}{G}\right)$ and the overall intensity $A = \log_2\sqrt{RG}$, are most commonly used to create the loess regression curve. Like the median normalization which shifts the median of each chip to zero, the loess normalization effectively shifts

the loess curve of the data to zero. The spatial 2D normalization fits a loess surface to the intensities at the x and y spot coordinates. This surface is then subtracted from the pre-normalized values to center the data. This procedure is often used separately on each print-tip group on a chip.

Caution

Loess normalization for two-channel data is computationaly intensive, and computations may take an extended amount of time.

Median Absolute Deviation Normalization (MAD)

Scale normalization attempts to align the variability of the expression intensity across chips. Yang et al (2001, 2002) suggest that for scale normalization a robust estimate such as median absolute deviation (MAD) may be used. For a collection of numbers x_1, ..., x_n, the MAD is the median of their absolute deviations from the median m=median$\{x_1,...,x_n\}$, MAD = median$\{ |x_1\text{-m}|, ... ,|x_n\text{-m}| \}$.

`maNormMain` Function

The main function for location and scale normalization of two-channel microarray data is `maNormMain`. Normalization is performed for each chip (independently) in a given batch of arrays using location and scale normalization procedures specified by the lists of functions `f.loc` and `f.scale`. Typically, only one function is given in each list, otherwise composite normalization is performed using the weights computed by the functions `a.loc` and `a.scale`. When both location and scale normalization functions (`f.loc` and `f.scale`) are passed, location normalization is performed before scale normalization. That is, scale values are computed for the location normalized log-ratios. `maNormMain` operates on an object of class `marrayRaw` (or possibly `marrayNorm`, if normalization is performed in several steps) and returns an object of class `marrayNorm`.

Default Normalization Parameters For `maNormMain`

`maNormMain` accepts any of the normalization methods listed in Table 6.2. The default parameters for these methods are also listed. The default normalization parameters can be changed by supplying the parameters as arguments in the normalization method call as follows:

```
# Two-channel normalization
# Default - Within-print-tip-group loess location
# normalization of first two arrays in the swirl
```

```
# dataset, but change default
# span parameter from .4 to .6
> swirl.norm <- maNormMain(swirl[,1:2],
  f.loc=list(maNormLoess(span=.6)))
```

Table 6.2: *Two-channel scale and location normalization methods performed through* `maNormMain`.

Normalization Method and Default Settings	**Description**
`maNormMed` Defaults: `x=NULL, y="maM", subset=TRUE`	Location normalization using the global median of intensity log-ratios for a group of spots.
`maNormLoess` Defaults: `x="maA", y="maM", z="maPrintTip", w=NULL, subset=TRUE, span=0.4`	Location normalization to a fitted loess curve (usually for M vs. A).
`maNormMAD` Defaults: `x=NULL, y="maM", geo=TRUE, subset=TRUE`	Scale normalization using the median absolute deviation (MAD) of intensity log-ratios for a group of spots.
`maNorm2D` Defaults: `x="maSpotRow", y="maSpotCol", z="maM", g="maPrintTip", w=NULL, subset=TRUE, span=0.4`	2D spatial location normalization. Normalizes to the smoothed intensity surface (loess surface) by print-tip group at each x, y coordinate.

Examples With `maNormMain`

Let's normalize the swirl data using a variety of methods in the `maNormMain` function. The normalization methods will be applied to the set of chips given. If you don't want to normalize across treatment conditions, then the `marrayRaw` objects can be subset as shown below.

```
# The swirl dataset. For description type ?swirl
# or help(swirl)
> swirl

# Global median normalization for arrays 81 and 82
#  (first two chips in set)
```

```
> swirl.normGmed <- maNormMain(swirl[,1:2],
  f.loc=list(maNormMed(x=NULL,y="maM")))

# Global median normalization over all chips in swirl
> swirl.normGmed <- maNormMain(swirl,
  f.loc=list(maNormMed(x=NULL,y="maM")))

# 2D spatial location normalization of array 93
> swirl.norm2D <- maNormMain(swirl[,3],
  f.loc=list(maNorm2D()))

# A normalization that is a weighted average
# of the loess normalization over the chip and
# the loess normalization over the print-tip groups
> swirl.norm <- maNormMain(swirl[,1],
  f.loc=list(maNormLoess(x="maA",y="maM",z=NULL, span=.5),
   maNormLoess(x="maA",y="maM",z="maPrintTip")),
   a.loc=maCompNormA())
```

Normalization Functions:

`maNorm`, `maNormScale`

Simple wrapper functions to `marrayNormMain` are provided by `maNorm` and `maNormScale`. These wrappers send default accessor methods and settings to `marrayNormMain` as outlined in Table 6.3 and Table 6.4. Two-channel normalization from the S+ARRAYANALYZER normalization dialog uses these functions and associated method names.

Table 6.3: *The `norm` parameter of `maNorm` results in the following normalization methods and settings being passed to `maNormMain`*

Normalization Method	f.loc Value	Summary
`median`	`f.loc = list(maNormMed(x = NULL, y = "maM", subset = subset))`	Median normalization by chip.
`loess`	`f.loc = list(maNormLoess(x = "maA", y = "maM", z = NULL, w = NULL, subset = subset, span = span))`	Normalization to loess curve of chip's M vs A.
`twoD`	`f.loc = list(maNorm2D(x = "maSpotRow", y = "maSpotCol", z = "maM", g = "maPrintTip", w = NULL, subset = subset, span= span))`	2D spatial location normalization. Normalizes to the smoothed intensity surface (loess surface) by print-tip group at each x, y coordinate.
`printTipLoess`	`f.loc = list(maNormLoess(x = "maA", y = "maM", z = "maPrintTip", w = NULL, subset = subset, span= span))`	Normalizes to the loess curve of M vs. A within each print-tip group on each chip in the object.
`scalePrintTipMAD`	`f.loc = list(maNormLoess(x = "maA", y = "maM", z = "maPrintTip", w = NULL, subset = subset, span= span)), f.scale = list(maNormMAD(x = "maPrintTip", y = "maM", geo = TRUE, subset = subset))`	Normalizes to the loess curve of M vs. A within each print-tip group, followed by within-print-group scale normalization using the median absolute deviation.

Table 6.4: *The norm parameter of maNormScale results in the following normalization methods and settings being passed to maNormMain*

Normalization Method	f.loc Value	Summary
globalMAD	f.loc = NULL, f.scale = list(maNormMAD(x = NULL, y = "maM", geo = geo, subset = subset))	Scale normalization over each chip using the median absolute deviation (MAD), this allows between slide scale normalization.
printTipMAD	f.loc = NULL, f.scale = list(maNormMAD(x = "maPrintTip", y = "maM", geo = geo, subset = subset))	Within-print-tip-group scale normalization using the median absolute deviation.

Examples With maNorm and maNormScale

Let's look at some examples using maNorm and maNormScale.

```
# scalePrintTipMAD performs both location and scale
# normalization.
> swirl.PrintTipMAD <- maNorm(swirl,
  norm="scalePrintTipMAD")

# print tip loess
> swirl.ptloess <- maNorm(swirl, norm="printTipLoess")

# globalMAD
> swirl.gMAD <- maNormScale(swirl, norm="globalMAD")
> swirl.gMADS <- maNormScale(swirl[,c(2,4)],
  norm="globalMAD")

# printTipMAD
> swirl.ptMAD <- maNormScale(swirl, norm="printTipMAD")
```

Normalization Between Arrays

After individual arrays have been normalized for differences in the red and green fluorescence intensities, additional normalization can be done between arrays. S+ARRAYANALYZER offers quantile normalization on the individual R/G channels as well as on the average of the two channels (A) through the GUI for this purpose. Normalization between arrays is needed if downstream analyses (e.g.

ANOVA) compare experimental conditions that vary between arrays. The between array normalization options are obtained by selecting a method for the **Between Array** field in the GUI. Refer to section Normalizing with the GUI and Chapter 4, Examples: Two-Color Data for more information and examples of normalizing through the GUI.

PRE-PROCESSING AND NORMALIZATION FOR AFFYMETRIX PROBE-LEVEL DATA

Affymetrix data typically arrives as **.DAT**, **.CEL** and .**CHP** files. The **.DAT** files contain the raw images as processed by the scanner. The **.CEL** files contain expression measures for each individual probe on the chip. The **.CHP** files contain summaries of the individual probe-level data for each gene/transcript. This section discusses methods for analyzing (correcting, summarizing and normalizing) the **.CEL** probe-level data. Examples of these procedures from the GUI can be found in Chapter , Examples: Affymetrix Probe-Level Data.

The key task is to convert probe-level data to one expression value for each gene/transcript which can then be used to test for differential gene expression. This is typically achieved through the following sequence of steps:

1. Exploratory data analysis and diagnostics.
2. Background correction.
3. Probe specific background correction (e.g. ,subtracting MM).
4. Normalization.
5. Summarizing the probe set values into one expression measure and, in some cases, a standard error for this summary.

As discussed in the section Workflow on page 217, normalization can be done before and/or after summarizing probe-level data.

Steps 2-5 above can be done using separate functions or together using functions such as `expresso`. These functions as well as functions for plotting probe-level data for exploratory data analysis are discussed in the next sections.

In S+ARRAYANALYZER, the `expresso` function provides many options to handle the tasks in steps 2-5 above. Examples are given in section Summarization in S+ARRAYANALYZER - Command Line on page 247.

CDF and Probe Libraries

In order to compute expression summaries and/or normalization of Affymetrix probe-level data you will need to have the Affymetrix CDF information available. DNA sequence information is required to use GC-RMA for background correction. In S-PLUS the information is provided in named lists stored in libraries. S+ARRAYANALYZER combines the Affymetrix CDF and probe libraries into one library called <*chipname*>cdf.

The complete installation of S+ARRAYANALYZER includes several libraries for common chips in the S-PLUS library directory. For example, hgu95av2cdf and hgu133acdf, hgu133bcdf, mgu74av2cdf, moe430acdf, rae230acdf, rgu34acdf. If you are working with these chips (hgu95av2, hgu133a, hgu133b, mgu74a, rae230a) then you do not need to do anything, the S+ARRAYANALYZER functions that operate on the CEL data will find the named lists. If you used the custom or standard install and you need to manually install cdf libraries for other chips, copy the cdf libraries into the S-PLUS library directory.

Each cdf library contains four objects:

- <chipname>cdf, derived from the Affymetrix Chip Definition File, gives the pm and mm information for each gene represented on the chip. This is used by a variety of summarization methods in the affy library.
- <chipname>probe, derived from the Affymetrix "probe_tab" files, gives the DNA sequence for each spot. This is used by the GC-RMA background correction method and by other functions in the matchprobes library.
- <chipname>.xy2i and <chipname>.i2xy are little functions that map from the x,y position on the chip to the linear index of a row in a CEL file.

S+ARRAYANALYZER functions need to access the named lists in these libraries when doing probe-level operations. If the lists are not available, S+ARRAYANALYZER attempt to load the needed library, if it cannot find the library an error or warning message will appear.

The CDF and probe set information for Affymetrix chips is available on the S+ARRAYANALYZER Web site:

http://www.insightful.com/support/ArrayAnalyzer/DataLibs/README.html

There is a CDF zip file for each chip and each zip file unpacks to create a library. These libraries include the probe set information listed above that is required by GC-RMA. Please refer to the section CDF and Probe Libraries for more details on loading additional libraries.

Affymetrix Diagnostic plots

Both MvA plots and box plots are available from the expression summary and normalization dialogs under the S+ARRAYANALYZER menu. Affymetrix data uses one treatment condition per chip. Comparisons can be made between treatments and within treatments. To compare expression *within* treatment conditions the *intensity log-ratio* (M) and the *average intensity* (A) are commonly defined as follows:

$$M = \log_2(E_{ki}/E_{kj})$$

$$A = \frac{\log_2(E_{ki}E_{kj})}{2}$$

where E_{ki} is the expression intensity for replication i, $i \neq j$; i, j =1,..., # of replications of the k-th treatment.

To compare expression *between* treatment conditions the *intensity log-ratio* (M) and the *average intensity* (A) are commonly defined as follows:

$$M = \log_2(E_{ki}/E_{lj})$$

$$A = \frac{\log_2(E_{ki}E_{lj})}{2}$$

where E_{ki} is the expression intensity for a given replicate and treatment; i =1,..., # of replications of the *k*-th treatment, and j =1,..., # of replications of the *l*-th treatment.

Box Plots

Since Affymetrix probe level data is on the raw scale, box plots from the GUI plot the $\log_2$ of the intensities. Please refer to section Box Plots on page 218 for a general discussion of box plots.

MvA plots

The function `mva.pairs`, or the MvA plots from the GUI, show pairwise graphical comparisons, e.g. between replicates of a treatment condition. The axes of these plots are the log-ratio intensities (M) between a replicate chip pair vs. the average log sum (A) intensities of the chip pair.

The pairwise scatter plots are shown on the top right half of the graph and the inter-quartile range (IQR) of the log ratios is shown on the bottom left half of the graphs. The chip labels are given on the diagonal. These plots can be particularly useful in diagnosing problems in replicate sets of arrays. Figure 6.5 shows an MvA plot for one treatment condition of the Dilution experiment (there are two replicates of this condition). Please refer to the help files for information about the Dilution dataset. Plots from the GUI show one

pairs plot per treatment condition. From the GUI a maximum of 2000 genes are plotted. If the chips have more than 2000 genes, then a random sample of 2000 genes are plotted.

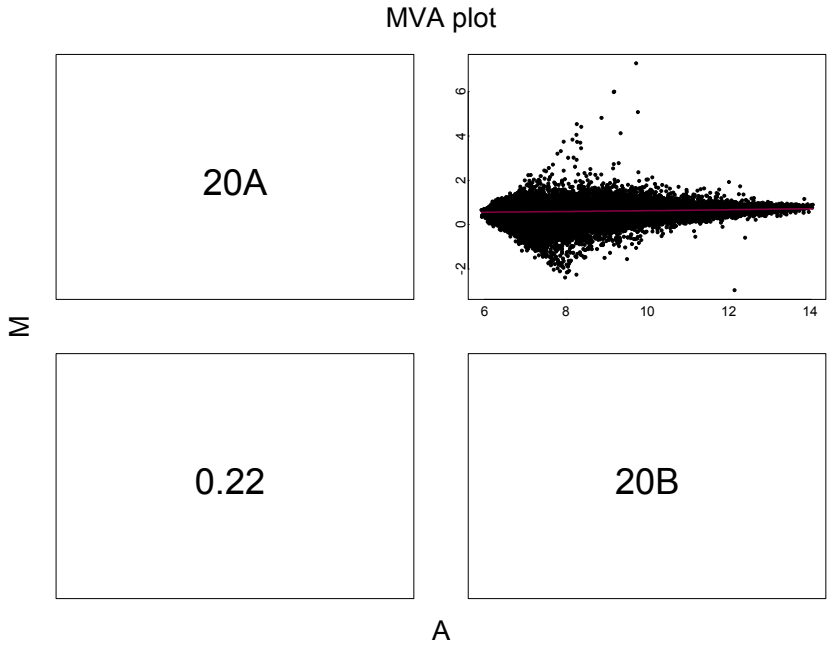


Figure 6.5: *MvA plot of one treatment group of Dilution experiment.*

Plots from the Command Line

Table 6.5 lists the diagnostic plots available in S+ARRAYANALYZER from the command line. The functions `box plot`, `hist`, and `image` are methods which work on `AffyBatch` objects (See the AffyBatch-class help file.). The input to `mva.pairs` is the matrix of expression measures (usually the log-intensity matrix is used).

Table 6.5: *Exploratory data analysis plots available from the command line for Affymetrix probe-level data.*

Function Name	Description
`boxplot`	Box plot of log base 2 of intensity matrix.
`hist`	(Calls `plotDensity`) Plots the non-parametric density estimates of the given matrix.
`mva.pairs`	MvA plots
`image`	Raw image plots - can be used to detect spatial artifacts.
`plotAffyRNAdeg`	(Requires object returned from `AffyRNAdeg`.) RNA degradation plots - aid in assessment of RNA quality.

Background Correction

Expression intensity measurements are summaries of the fluorescence intensities for the pixels contained within each chip spot. The background of the chip contributes to this signal and the background noise levels may not be consistent over the chip. Background correction aims to quantify and subtract this background signal from the expression intensities. S+ARRAYANALYZER provides four methods for correcting Affymetrix probe-level chips for background signal and inconsistencies. Three of these methods are available through the function `bg.correct`. The fourth method, GC-RMA, is a separate function, `gcrma`.

Methods via bg.correct

The background correction methods available with `bg.correct` can be obtained by typing

```
> bgcorrect.methods
[1] "mas"  "none" "rma" "rma2"
```

`rma and rma2`

The `rma` background adjustments assume the PMs are a convolution of the normal and exponential distributions. According to Bolstad (2002a) we can write this as O=S+N where N is the background and S is the signal. It is assumed that S is distributed exp(α) and N is distributed N(μ,σ^2). The background corrected PM values returned for each chip in the object are then E[s|O=o]. This expectation is equal to

$$a + \frac{b\left[\phi\left(\frac{a}{b}\right) - \phi\left(\frac{(o-a)}{b}\right)\right]}{\left[\Phi\left(\frac{a}{b}\right) + \Phi\left(\frac{(o-a)}{b}\right) - 1\right]}$$

where $a = s - \mu - \sigma^2\alpha$, $b = \sigma$, and ϕ and Φ are the normal density and cumulative distribution function respectively.

Caution

The `rma` methods adjust the PM values but leaves the MM values intact. This is problematic if a PM correction is done after the background correction using MM values which have not been background corrected.

`rma` and `rma2` differ only in how they estimate a set of parameters. `rma2` is only available for the command line and is not recommended. The `rma2` method is available via `bg.correct.rma2(object, bgtype=1)` or `bg.correct(object, method="rma2")`. Setting `bgtype=2` will result in the original rma method being applied.

`mas`

The `mas` background method performs the noise correction described in the "Statistical Algorithms Description Document (SADD)", a white paper from Affymetrix. This method divides the chip into a given number of zones and uses the lowest 2% of the intensity values

to compute the background intensity within each zone. Smoothing across zones is done by computing a zone weight which is based on the distances of spots to zone centers. The background at each cell location (x,y) is computed using these weights. A similar computation is made for the noise at each cell.

The background corrected value is computed as a function of the background at (x,y), noise at (x,y), and the threshold and floor noise values at each (x,y) cell location (based on the noise at (x,y)) such that the cell intensity remains positive.

An Example With `bg.correct`

`bg.correct` takes an `AffyBatch` object and returns an `AffyBatch` object. Following are some examples of background correcting a sample extracted from the Dilution experiment. Please refer to the Dilution help file for more details on the data.

```
> tmp <- bg.correct(Dilution, method= "mas")
> tmp <- bg.correct.rma(Dilution)
> tmp <- bg.correct.mas(Dilution)
```

Background Correction with `gcrma`

GC-RMA is a robust multi-array expression measure using sequence information. It provides background estimates based on a model using GC content. GC-RMA is a modified version of RMA that models intensity of probe level data as a function of GC-content. The theory being that you would expect to see higher intensity values for probes that are GC rich due to increased binding. This is seen as an improvement over RMA which does not consider the physical processes.

`gcrma` calls the `rma` function after background correction. Therefore, expression intensities returned from `gcrma` are normalized and summarized using quantile normalization and median polish summarization. The function accepts an `AffyBatch` class object and returns and `exprSet` class object.

PM Correct Methods

One may wish to correct the PM intensities in a `ProbeSet`[1] for *non-specific binding*, hybridization that occurred at random. Affymetrix chips provide a mechanism for measuring non-specific binding

1. A `ProbeSet` object is the collection of cloned PM and MM spot intensities for one gene.

through the mismatch probes (MM). The amount of binding that occurs at these spots is a measure of the amount of random binding that is occurring in the experiment.

The available methods can be obtained by typing

```
> pmcorrect.methods
[1] "mas"        "pmonly"     "subtractmm"
```

`mas` and `subtractmm` are the pm correction methods performed by Affymetrix MAS 4.0 (`subtractmm`) and MAS 5.0 (`mas`) software.

`subtractmm` returns the difference between the PM and MM intensity values. This can lead to negative values for the intensity.

`pmonly` returns the PM intensity values from the `ProbSet` PM slot.

`mas` correction allows for the possibility that the MM intensity is larger than the PM intensity for a particular probe pair within a probe set. The `mas` method is described in the Affymetrix "Statistical Algorithms Description Document (SADD)" available from Affymetrix.

An Example with `pmcorrect`

The input to the `pm.correct` functions can be either a `ProbeSet` or `AffyBatch` object and the return value is a matrix of corrected PM values for each chip in the input object. An object of class `ProbSet` contains the PM and MM data for a probe set from one or more samples. `ProbeSet` objects can be created by applying the method `probeset` to instances of `AffyBatch`. We illustrate the procedure using the example data, `affybatch.example`, in the affy library data directory. This data set gives a subset of the values read from a HU6800 CEL file.

```
> pps <- probeset(affybatch.example,
          geneNames(affybatch.example)[1:2])[[1]]
> pps.subtractmm <- pmcorrect.subtractmm(pps)
```

If no subsetting is desired we can simply use the `AffyBatch` object in the correction procedure.

```
> pmCor.mas <- pmcorrect.mas(affybatch.example)
```

We can replace the original PM values with the corrected PM values, by typing

```
> affybatch.example.tmp <- affybatch.example
> pm(affybatch.example.tmp) <- pmCor.mas
```

Normalization

Like two-channel arrays, the spot intensities on Affymetrix arrays include variations due to sample preparation, manufacturing of the arrays and array processing (labeling, hybridization, and scanning). Many researchers have pointed out the need for normalizing Affymetrix arrays. See for example Bolstad et al. (2002) and Irizarry et al. (2003a). S+ARRAYANALYZER provides a variety of normalization methods for cell-level data.

Location normalization methods:

- constant
- contrasts
- invariantset
- loess

Scale normalization methods:

- qspline
- quantiles
- quantiles.robust
- vsn

The main function for normalizing `AffyBatch` objects is `normalize`. The `normalize` function accepts `AffyBatch` objects and returns `AffyBatch` objects. `AffyBatch` objects store the experimental information about the probe level data. Please refer to the affy library documentation (**splus62/library/affy/affy.pdf**) or the `AffyBatch-class` and `exprSet-class` help files for more details.

`normalize` Function

The `normalize` function is a generic wrapper which calls the `normalize.AffyBatch.(method)` functions. These functions extract the intensity columns for the given set of chips and pass each intensity column (or matrix, depending on whether the normalization is over a chip or across a group of chips) to the functions `normalize.(method)`. The `normalize.(method)` functions can be called directly by the user by passing in the appropriate intensity vector or matrix.

You can obtain a list of normalization methods for an object by typing

```
> normalize.methods(Dilution)
```

This list of methods for `AffyBatch` objects can also be obtained by typing

```
> normalize.AffyBatch.methods
[1] "constant"         "contrasts"        "invariantset"
[4] "loess"            "qspline"          "quantiles"
[7] "quantiles.robust" "vsn"
```

The normalization methods available in S+ARRAYANALYZER for Affymetrix CEL files (AffyBatch objects) are shown in Table 6.6.

Table 6.6: *Normalization methods available through the `normalize` function.*

Normalization Methods	Default Function Values	Description
Location Normalization Methods		
`constant`	`ref =1, FUN = mean na.rm = T`	Normalizes one chip to a have a given mean or median value or normalizes a set of chips (if the object is an AffyBatch object) to have the same mean or median as a given reference chip.
`contrasts`	`span = 2/3, choose.subset = T, subset.size = 5000, verbose = T, family = "symmetric"`	Performs a modified loess normalization using contrasts to create a linear combination of all pairwise combinations of chips.
`invariantset`	`prd.td = c(0.003, 0.007)`	The chip with the median mean intensity for the set is chosen as the reference chip. Based on the rank of the intensities, a group of invariant genes is chosen for each chip. A smooth spline is fit to this invariant set of genes. This is a pairwise normalization for each chip in the set to the reference chip.

Table 6.6: *Normalization methods available through the* `normalize` *function.*

Normalization Methods	**Default Function Values**	**Description**
`loess`	`subset = sample(1:(dim(mat)[2]), 5000), epsilon = 10^2, maxit = 1, log.it = T, verbose = T, span = 2/3, family.loess = "symmetric"`	Normalizes the chips with respect to each other by forcing log ratios to be scattered around the same constant curve. This is accomplished on more than two arrays by averaging the pairwise loess curves.
ScaleNormalization Methods		
`qspline`	`target = NULL, samples = NULL, fit.iters = 5, min.offset = 5, spline.method = "natural", smooth = TRUE, spar = 0, p.min = 0, p.max = 1., incl.ends = TRUE, converge = FALSE, verbose = TRUE, na.rm = FALSE`	The quantiles from each array and the target are used to fit a system of cubic splines to normalize the data.
`quantiles`		Assuming an underlying common distribution, the set of chips are normalized so that their quantiles have the same value.
`quantiles.robust`	`x,weights = NULL, remove.extreme = "variance", n.remove=1, approx.meth = FALSE`	Quantile normalization with options to: Eliminate chips with high variability. Eliminate chips with means too disparate from others. Down weight particular chips in the computation of the mean.
`vsn`	`subsample = 20000, niter = 4`	Variance stabilizing normalization.

An Example with `normalize`

Below we use the melanoma data set (Fox et al. (2001)) to demonstrate various normalization procedures. The melanoma dataset is discussed in section Melanoma Data on page 69. We first read in the data. This can be done through the GUI as shown in section Importing Data on page 95 of Chapter , Examples: Affymetrix Probe-Level Data.

```
> directory<-paste(getenv("SHOME"),"module/ArrayAnalyzer/
examples",sep="/")
> cgnames <- paste(directory,
c("cg2a.CEL","cg2b.CEL","cg24a.CEL", "cg24b.CEL"), sep="/")
> NCImelanoma <- ReadAffy(filenames=cgnames)
```

The data should be corrected for specific binding and background noise. One way to do this is to simulate the Affymetrix MAS 5.0 software as follows:

```
# correct melanoma CEL data
# background correct
> NCImelanoma <- bg.correct(NCImelanoma, method="mas")

# Correct using MM as controls
> tmp <- pmcorrect.mas(NCImelanoma)

# Add the correct PM values back into melanoma object
> pm(NCImelanoma) <- tmp
```

The default method for `normalize` is quantiles. In this next example we subset the `AffyBatch` object by treatment (0 and 24 hours), normalize each subset, then merge the objects into a single, normalized `AffyBatch` object.

```
> mel.norm.quantiles  <-
   merge(normalize(NCImelanoma[1:2]),
   normalize(NCImelanoma[3:4]))
```

We can normalize each replicate set to the median of one of the chips by typing

```
> mel.norm.constant  <-  merge(normalize(NCImelanoma[1:2],
  method="constant"), normalize(NCImelanoma[3:4],
  method="constant" ))
```

Arguments to the normalization methods can be passed through normalize as optional arguments. This example normalizes all four chips in the melanoma experiment.

```
> mel.norm.loess <- normalize(NCImelanoma,
  method="loess", span=.5)
```

The corrections and normalization can be done in one step using the `expresso` function. This function also summarizes the probe-level data. The resulting object is of class `exprSet`.

```
> melanoma.exprSet <- expresso(NCImelanoma,
  bgcorrect.method="mas", pmcorrect.method="mas",
   normalize.method="constant", summary.method="mas")
```

Summarization Methods

Affymetrix and some other high density oligonucleotide arrays include multiple spots per gene/transcript. In Affymetrix arrays there are 11, 16 or 20 probe pairs in a probe pair set, with each probe pair consisting of a perfect match (PM) and mismatch (MM) probe. The oligos are 25-mers and the MM probe uses the Watson-Crick complement at the 13th position.

A key data operation is the summary of the 11-20 probe pair set intensities into a single value for each gene/transcript that faithfully represents the expression of that gene/transcript. The Affymetrix MAS4.0 software did a poor job at this summarization, by simply taking the average difference of the PM and MM values for each probe pair set. Affymetrix MAS5.0 software does a better job of this summarization; this is described below. Several other summary methods for probe pair sets have emerged most notably those of Li and Wong (2001) and Irizarry et al. (2003b). This is an active area of research and as stated by Parmigiani et al (2003), "there is mounting evidence that alternative summarization to the defaults currently implemented by Affymetrix may provide improved ability to detect biological signal."

The available summary methods can be obtained by typing

```
> express.summary.stat.methods

[1] "avgdiff" "liwong" "mas" "medianpolish" "playerout"
```

Note that `avgdiff` and `mas` methods refer to the methods described in the Affymetrix manual versions 4.0 and 5.0 and the Affymetrix "Statistical Algorithms Description Document (SADD)" available from Affymetrix.

`avgdiff` The `avgdiff` method implements an approach similar to that of the MAS4.0 software. This involves forming the differences PM-MM for each probe pair; calculating the mean and standard deviation (sd) of these differences, removing pairs with a difference of greater than 3 standard deviations from the mean; and recalculating the mean from the trimmed set.

`liwong` The `liwong` method fits the model described in LI and Wong (2001a, 2001b). The default setting gives the current PM-only default. The reduced model (previous default) can be obtained using `pmcorrect.method="subtractmm"`.

`mas` The `mas` method implements an approach similar to that of the MAS5.0 software. This includes forming the differences PM-MM for each probe pair and then condensing these within a probe pair set in a "robust" manner. Outlier probe pairs are not dropped as in the `avgdiff` calculation, they are down-weighted. The median of the probe pair differences within a probe pair set is calculated, and each probe pair difference is down-weighted as a function of its distance from the median. The probe pair differences are then combined in a one-step version of the Tukey biweight procedure.

`medianpolish` The `medianpolish` algorithm works by alternately removing the row and column medians, and continues until the proportional reduction in the sum of absolute residuals is less than `eps` or until there have been `maxiter` iterations. In combination with the `bg.correct.rma` background correction method and the `quantiles` normalization method, this forms the RMA probe-level analysis method of Irizarry et al. (2002, 2003b).

`playerout` The `playerout` method computes a weighted mean of the PM values, based on the method described by Lazaridris et al. (2002). Our tests show this method gives unstable results. This method is not recommended.

Summarization in S+ARRAYANALYZER - Command Line

Summarization in S+ARRAYANALYZER can be done through the **Affymetrix Expression Summary** dialog as demonstrated in Chapter 3, Examples: Affymetrix Probe-Level Data. From the command line, summarization can be done as a separate step using the wrapper function `computeExprSet` on an `AffyBatch` object or through the functions `generateExprVal.method.(method)`. These functions require a matrix of probe intensities with rows representing probes and columns representing samples. Details of using these functions can be found in the help files (for example type

```
> ?generateExprVal.method.avgdiff).

# computeExprSet can be used to summarize probe data
> affybatch.example
> ids <- c("A28102_at","AB000114_at")
> eset <- computeExprSet(affybatch.example,
  pmcorrect.method="pmonly",
  summary.method="avgdiff", ids=ids, warnings=FALSE)

# generateExprVal.method.(method) requires an
# intensity matrix
> probes <- pm(SpikeIn)
> medianpolish <-
      generateExprVal.method.medianpolish(probes)
```

The correcting, normalization and summarization steps can be done in one step using the `expresso` functions. Details on `expresso` can be found in the help file, invoked by entering

```
> ?expresso
```

Summarization Examples

In these examples we correct the data for background signals and noise, normalize the data at the probe-level, and summarize the probe-level data into one value per gene/transcript. We do this all using the `expresso` function.

Affymetrix MAS 5.0

To obtain a summary similar to MAS 5.0 use

```
> eset <- expresso(affybatch.example, normalize=FALSE,
  bgcorrect.method="mas",
  pmcorrect.method="mas",
```

```
    summary.method="mas")
> eset <- affy.scalevalue.exprSet(eset)
```

Notice that, in this case, we normalize after we obtain summarized expression measures. The function `affy.scalevalue.exprSet` performs a normalization similar to that described in the MAS 5.0 manual (see the section on `affy.scalevalue.exprSet` on page 253). This is a simple global scaling in which the user enters a target value (TGT value). The average signal (across all probes on each chip) is calculated for each chip and a scale factor (SF) is determined for each chip such that chip.mean*SF = TGT. Thus the signals on each chip are scaled by a single number for each chip - a crude form of normalization.

Li and Wong (2001) MBEI

To obtain a probe-level normalized summary similar to Li and Wong's MBEI one can use (This is computationaly intensive.)

```
#> eset <- expresso(Dilution,
# normalize.method="invariantset",
# bg.correct=FALSE,
# pmcorrect.method="pmonly",
# summary.method="liwong")
```

This gives the current PM-only default. The reduced model (previous default) can be obtained using `pmcorrect.method="subtractmm"`.

RMA method of Irizarry et al. (2002)

The RMA method of Irizarry et al. (2002) can be obtained using `expresso` as follows:

```
> eset <- expresso(affybatch.example,
  normalize.method="quantiles",
  bgcorrect.method="rma",
  pmcorrect.method="pmonly",
  summary.method="medianpolish")
```

Equivalently, the `rma` function can be used and is faster for this series of operations.

```
> eset <- rma(affybatch.example)
```

Summarization in S+ARRAYANALYZER - GUI Dialog

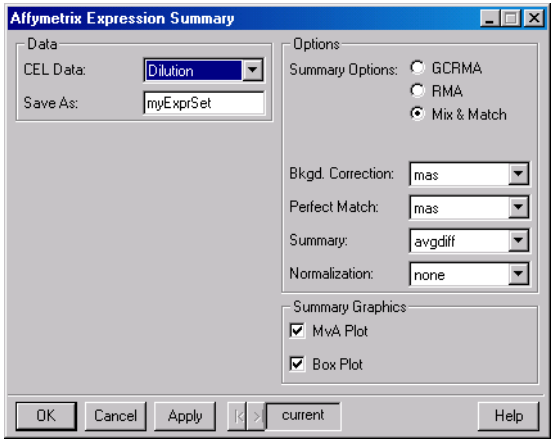


Figure 6.6: ***Affymetrix Expression Summary*** *dialog*

The **Affymetrix Expression Summary** dialog provides a convenient way to transform raw probe-level data into data ready to be tested for differential expression. This dialog provides options for background correction, probe specific background correction (subtracting MM), normalization and summarization of the probe sets. The input to this dialog is an `AffyBatch` class object created from the import dialog or at the command line. The output is an object of class `exprSet` and graphical representation of the resulting summarized data. The graphical output is discussed in section Affymetrix Diagnostic plots on page 234.

Summarization and Correction Options

This dialog allows you to choose either **GC-RMA**, **RMA** or to **Mix & Match** correction, summarization and normalization methods. GC-RMA, as discussed in section Background Correction with `gcrma` on page 239 is primarily a background correction method. From this dialog, however, the **GC-RMA** option also normalizes the arrays by `quantiles` and summarizes the probe sets using `medianpolish`. Choosing **RMA** will return data that has been corrected using `rma` background correction, `pmonly` PM correction, `quantiles` normalization and `medianpolish` to summarize the probe sets.

The **Mix & Match** option allows you to be very creative in how the final summarized data is created. However, not all combinations of techniques make sense. It is up to the user to know which options make sense to use at the same time.

NORMALIZATION METHODS FOR AFFYMETRIX MAS DATA

Affymetrix data typically arrives as **.DAT**, **.CEL** and **.CHP** files. The **.DAT** files contain the raw images as processed by the scanner. The **.CEL** files contain expression measures for each individual probe on the chip - analysis of these probe-level data is described in Chapter 3, Examples: Affymetrix Probe-Level Data and in this chapter in section Pre-Processing and Normalization for Affymetrix Probe-Level Data. The **.CHP** files contain summaries of the individual probe-level data for each gene/transcript. Analysis of these summarized data is described in this section. These data have been background adjusted and summarized into a single expression value per gene/transcript using the Affymetrix MAS software.

Affymetrix version 5.0 software has adjusted the probe-level intensity values as follows:

- Global background signal and noise have been subtracted and thresholded as described in the Affymetrix "Statistical Algorithms Description Document (SADD)" available from Affymetrix. This method is also described in section `mas` on page 238.
- The 11-20 mismatch (MM) and perfect match (PM) values have been summarized using a Tukey biweight procedure as described in the Affymetrix document "Statistical Algorithms Description Document (SADD)" and section `mas` on page 246.
- If requested by the user, the software scales the signal using a trimmed mean (2% of the data at either end is trimmed away before the mean is computed.)

The output intensity for MAS 5.0 data is termed *Signal*

Affymetrix version 4.0 software adjusts the probe level data by:

- Subtracting the global background signal and noise as described in section `mas` on page 246.
- Summarizing the 11-20 mismatch (MM) and perfect match (PM) values using a simple trimmed average difference procedure (see section `avgdiff` on page 246).

The output intensity for MAS 4.0 data is termed *Avg Diff*.

The summarized Affymetrix data from both MAS4 and MAS5 have not been suitably normalized for differential expression testing. Note that the MAS software allows a very simple global scaling in which the user enters a target value (TGT value). With this method the average signal (across all probes on each chip) is calculated for each chip and a scale factor (SF) is determined for each chip such that chip.mean*SF = TGT. Thus the signals on each chip are scaled by a single number for each chip - a crude form of normalization. S+ARRAYANALYZER provides three methods for normalizing this summarized data from the command line, `medianIQR`, `vsn` and `affy.scalevalue.exprSet`. Quantile normalization is also available from the GUI.

Normalization Methods

`Dilution.exprSet` is a sample `exprSet`[1] object available in the S+ARRAYANALYZER database. `Dilution.exprSet` is a summarized version of the Dilution experiment object, `Dilution`. Please refer to the help files for more details. We use `Dilution.exprSet` to demonstrate normalization of summarized microarray data.

medianIQR

`medianIQR` normalization scales the summarized chip data so that they have the same inter-quartile range as the maximum IQR for the set and the median of each chip's data is shifted to the maximum median of the chip set. `medianIQR` takes as input an expression intensity matrix (each column is one chip's values) and returns a matrix of the same dimensions - one column for each chip in the set.

`medianIQR` can be used from the command line as follows:

```
# Normalizing each treatment group separately
> DilutionEsetNormTmt1 <-
   medianIQR.norm(Dilution.exprSet[,1:2]@exprs)

> DilutionEsetNormTmt2 <-
   medianIQR.norm(Dilution.exprSet[,3:4]@exprs)
```

1. An object of class `exprSet` contains information for experiments where the probe-level data has already been summarized into one expression value for each gene. Please refer to the Biobase documentation for more details (**splus62splus62/library/Biobase/Biobase.pdf**) or the `exprSet`-class help file.

The data can be plotted by typing the following:

```
# pre-normalized data box plot
# log transform the data for nicer plots
> boxplot(data.frame(log2(Dilution.exprSet@exprs)),
   ylim=c(0,15), style.bxp ="att")
# post-normalized data
> boxplot(data.frame(log2(
  cbind(DilutionEsetNormTmt1,DilutionEsetNormTmt2))),
  style.bxp ="att", ylim=c(0,15))
```

Note

When creating box plots from the normalization dialog, the log-intensity is used on the y-axis.

To continue working with an `exprSet` object, we can create a new `exprSet` object which has the normalized intensity information.

```
# normalize the data without subsetting
> DilutionEsetNorm.matrix <-
   medianIQR.norm(Dilution.exprSet@exprs)

# create new exprSet object with normalized intensities
> DilutionEsetNorm <- Dilution.exprSet
> DilutionEsetNorm@exprs <- DilutionEsetNorm.matrix
```

`affy.scalevalue.exprSet`

`affy.scalevalue.exprSet` shifts the mean intensity value of the chips to the same specified point. The default reference value is 500. The function accepts `exprSet` objects and returns an `exprSet` object.

Similar to `medianIQR`, `affy.scalevalue.exprSet` can be used to normalize summarized data as follows:

```
# Normalizing with affy.scalevalue.exprSet
> DilutionEsetScaleTmt1 <-
  affy.scalevalue.exprSet(Dilution.exprSet[,1:2],sc=100)
> DilutionEsetScale <-
  affy.scalevalue.exprSet(Dilution.exprSet, sc=100)
```

`vsn`

`vsn`, variance stabilizing normalization, is available from the GUI and the command line. This function can operate on `exprSet` objects and returns an `exprSet` object.

```
# Normalizing summarized data with vsn
> vsn(Dilution.exprSet)
> vsn(Dilution.exprSet[,1:2])
```

Diagnostic Plots for Summarized Affymetrix Data

MvA and box plots for Affymetrix summarized data are available through the Normalization dialog of S+ARRAYANALYZER GUI menu. An example of an MvA plot can be seen in section Affymetrix Diagnostic plots on page 234. Other plots such as histograms and qqplots can be obtained for any summarized data (data of class exprSet) by extracting the `exprs` slot values from the `exprSet` object. We demonstrate this using the previously `medianIQR` normalized data. The expression values are extracted, and the $\log_2$ transform is taken before the box plot is created.

```
> par(mfcol=c(1,2)) # two box plots on one page
> boxplot(data.frame(log2(Dilution.exprSet@exprs)),
  ylim=c(0,15), style.bxp = "att")
> boxplot(data.frame(log2(cbind(DilutionEsetNormTmt1,
   DilutionEsetNormTmt2))), style.bxp ="att", ylim=c(0,15))
```

From the plots in Figure 6.7 we can see that after normalization there are differences in the average expression levels between the two treatment groups.

Figure 6.7: *Before and after normalization box plots of the summarized Dilution dataset.*

REFERENCES

Affymetrix. (2002). *Statistical Algorithm Description Document.* Affymetrix, Santa Clara, CA.

Affymetrix. (2001). *Affymetrix MicroArray Suite, Version 5.0 User's Guide.* Santa Clara, CA.

Bolstad, B. M. (2002a) Comparing the effects of background, normalization and summarization on gene expression estimates. Unpublished Manuscript.

Bolstad, B. M. Irizarry, R. A., Astrand M. and Speed T. P. (2002). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* 19(2): 185-193.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829-836.

Dudoit, S. and Yang Y. H. (2003). Bioconductor R packages for exploratory analysis and normalization of cDNA microarray data. In *The Analysis of Gene Expression Data: Methods and Software.* G. Parmigiani, E. S. Garrett, R. A. Irizarry, and S. L. Zeger, editors. Springer, New York.

Dudoit, S.,Yang Y. H., Callow, M. J. and Speed, T. P. (2002). Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Statistica Sinica*, 12(1): 111-139. 3,4.

Fox J.W., Dragulev B., Fox N., Mauch C. and Nischt R. (2001). Identification of ADAM9 in human melanoma: Expression, regulation by matrix and role in cell-cell adhesion. *Proceedings of International Protelysis Society Meeting.*

Huber, W., Heydebreck, A., Sueltmann H.,Poustka, A., Vingron, M. (2002). Variance stabilization applied to microarray data calibration and to the quantification of differential expression, *Bioinformatics* 18 Suppl.1 S96-S104.

Huber, W., Heydebreck, A., Sueltmann H.,Poustka, A., Vingron, M. (2003). Parameter estimation for the calibration and variance stabilization of microarray data. *Statistical Applications in Genetics and Molecular Biology*. Vol. 2 No. 1, Article 3.

Irizarry, R. A., Gautier, L., and Cope, L. P. (2003a). Analysis of Affymetrix Prove-level Data. In *The Analysis of Gene Expression Data: Methods and Software*. Edited by G. Parmigiani, E. S. Garrett, R. A. Irizarry and S. L. Zeger. Published by Springer Verlag, New York.

Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U., Speed, T. P. (2003b). Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data. Accepted for publication in *Biostatistics*.

Irizarry, R. A., et. al. (2003) Exploration, normalization and summaries of high density oligonucleotide array probe level data. *Biostatistics* 4:249-264.

Irizarry, R. A., Bolstad B. M., Collin, F., Cope, L. M., Hobbs, B., and, Speed, T. P. (2002). Summaries of Affymetrix GeneChip Probe Level Data. *Nucleic Acids Research*. Vol. 31, No. 4 e15

Lazaridis, E., Sinibaldi, D., Bloom, G., Mane, S. and Jove, R. (2002). A simple method to improve probe set estimates from oligonucleotide arrays, Mathematical Biosciences, Volume 176, 1: 53-58

Lee J. K. and O'Connell M. (2003). An S-PLUS library for the analysis of differential expression. In *The Analysis of Gene Expression Data: Methods and Software*. Edited by G. Parmigiani, E. S. Garrett, R. A. Irizarry and S. L. Zeger. Published by Springer, New York.

Li, C., Wong, W. (2001a). Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection. Proceedings of the National Academy of Science U S A 98:31-36.

Li, C. and Wong, W. (2001b) Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application, Genome Biology 2(8): research0032.1-0032.11.

Parmigiani, G., Garrett, E. S., Irizarry, R. A., and Zeger, S. L. (2003). The analysis of gene expression data: an overview of methods and software. In *The Analysis of Gene Expression Data: Methods and Software*. Edited by G. Parmigiani, E. S. Garrett, R. A. Irizarry and S. L. Zeger. Published by Springer, New York.

S-PLUS 2000 Guide to Statistics, Volume 1, Data analysis Products Division, MathSoft, Seattle, WA.

Yang, Y. H., Dudoit, S., Luu, P., Lin, D. M., Peng, V., Ngai, J. and Speed, T. P. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 20(4).

Yang, Y. H., Dudoit, S., Luu, P., and Speed, T. P. (2001). Normalization for cDNA microarray data. In M. L. Bittner, Y. Chen, A. N. Dorsel, and E. R. Dougherty, editors, Microarrays: *Optical Technologies and Informatics*, volume 4266 of *Proceedings* of SPIE. May 2001.

Yang, Y. H., Dudoit, S. (2003). Bioconductor's marrayNorm Package. Bioconductor marrayNorm library documentation. January 23, 2003. p.3.

Wu, Z., Irizarry, R. A., Gentleman, R., Murillo, F. M., and Spencer, F. (2003). A Model Based Background Adjustment for Oligonucleotide Expression Arrays. Johns Hopkins University, Dept. of Biostatistics Working Papers.

DIFFERENTIAL EXPRESSION TESTING

7

Introduction **260**

Statistical Tests **261**
Within-Gene Two-Sample Comparisons 261
Local Pooled-Error Test 262
Raw P-Values 263

Controlling Type I Error Rates **265**
Controlling The False Positive Rate 266

GUI for Two Sample Testing **270**
Two Sample Dialog Input 270

GUI for LPE Testing **276**
LPE Testing Dialog Input 276

GUI for ANOVA Testing **280**
ANOVA Testing Dialog Input 280

Differential Expression Analysis Plots **286**
Common Plots 286

Differential Expression Summary Table Output **294**
Top 15 List 294
Complete Gene List 294

References **298**

INTRODUCTION

Differential expression testing in S+ARRAYANALYZER is defined as *statistical* testing of the difference in expression intensities between the treatment conditions under study. Effectively, this means that a separate hypothesis test is computed for each gene or probe on the chip. In two sample problems (e.g., two tissue types, treatment vs control), this boils down to a t-test (or other two sample test) for each gene. When computing tests for chips with many probes, setting a usual Type I Error (false positive) rate for *individual* tests will result in many false positives.

One key ingredient to good expression testing is controlling the family-wise error rate (FWER) or false discovery rate (FDR). There is a rich historical literature on this topic in statistical journals and texts (see, for example, Hochberg and Tamhane (1987), Westfall and Young (1993) or Hsu (1996)). The net effect of poor FWER control is wasted time and money in discovering many genes that really aren't differentially expressed. This topic is discussed in more detail in the section Controlling The False Positive Rate.

Another key ingredient to good statistical testing is obtaining good estimates of the standard error of differential expression for each gene. In some studies (e.g., with few replicates), specialized methods may be required to improve the power of the statistical test. We describe one approach to doing this in the section GUI for LPE Testing.

STATISTICAL TESTS

The S-PLUS environment is rich in methods for statistical modeling and hypothesis testing. Virtually all the traditional modeling and testing methodology is available in S-PLUS through either its GUI or its Command line, and many through both. Furthermore, because of the ease of programming S-PLUS, many new methods quickly find their way into the S-PLUS environment. This makes S-PLUS ideal for doing microarray analysis where many traditional methods (e.g., t-test, Wilcoxon test, ANOVA) are used but where the advantage of using cutting-edge methods (`loess` normalization, invariant set normalization, local pooled-error test) may provide a big pay-off by reducing false positives and negatives.

The focus of this chapter is on methods primarily supported through the GUI for differential expression testing. However, there are many techniques not covered here that are accessible through other sections of the GUI or through the Command line. See the Chapter 8, Cluster Analysis for examples on clustering and mixed effects models.

Within-Gene Two-Sample Comparisons

The focus of the S+ARRAYANALYZER GUI is two-sample problems. For two-sample problems it is quite easy to do the following:

1. Read the data
2. Summarize (probe-level Affy data)
3. Normalize
4. Test differential expression, and
5. Annotate differentially expressed genes from the GUI.

The within-gene two-sample comparisons implemented through the GUI include the following methods:

- **paired.t**: Paired t-test
- **t**: Welch's t-test (unequal variance)
- **t.equalvar**: Student's t-test (equal variance)
- **wilcoxon**: Wilcoxon signed-rank sum (non-parametric) test
- **t-permute**: Welch's t-test null distribution (and p-value) estimated by permutation

- **t.equalvar-permute**: Student's t-test with null distribution (and p-value) estimated by permutation.
- **wilcoxon-permute**: Wilcoxon signed-rank test with null distribution (and p-value) estimated by permutation.

The method names listed in bold are used in the GUI for specifying a particular testing method. All the basic methods (paired t, Welch's t, student's t, wilcoxon) are described in standard introductory statistical textbooks, such as Moore and McCabe (1999) or Snedecor and Cochran (1980).

The **permute** versions of the test procedures are based on permuting the intensity scores across treatment conditions repeatedly, re-computing the test statistic each time, to form a null distribution of the test statistic. The p-value is then obtained by quantifying the frequency of seeing a test statistic as extreme or more so than the one observed for the data. Using permutation methods for reasonable samples sizes (10 or more per experimental condition) can produce more accurate p-value estimates for data which may not satisfy the assumptions of the test procedure. In particular, tests for skewed intensity values may benefit from computing p-values by permutation rather than from the theoretical *(symmetrical)* distribution.

Cautionary Note

The **permute** versions of the tests should be used with caution for low replicate studies since the p-values are based on the total number of *possible* test statistics for permuted data. For example, in a two-sample study with two replicates (four arrays) the total number of different test statistics is at most six. This means that the *smallest possible* p-value for a two-sided alternative is 0.333! When the number of replicates increases to 10 per sample the minimum p-value drops to 0.00001 and when there are 20 arrays per sample the minimum p-value drops to 0.00000000001 (10^{-11}).

Local Pooled-Error Test

The local pooled-error (LPE) test is an experimental procedure designed for low replicate studies. When there are few replicates in a study, the degrees of freedom for estimating the standard error of differential expression within-genes may be as low as one or two. In this context, estimates of within-gene standard errors are imprecise, resulting in increased Type I and Type II errors. In particular, with the large number of genes on the chip, there will always be genes with low within-gene error estimates by chance, so that some signal-to-

noise ratios will be large regardless of mean expression intensities and fold-change. The local pooled error test attempts to avert this by combining within-gene error estimates with those of genes with similar expression intensity. In this sense, the LPE approach is similar to the SAM method of Tusher et al. (2001) and the B-statistic of Lonnstedt and Speed (2002).

LPE estimates used for differential expression testing are formed by pooling variance estimates for genes with similar expression intensities.

The LPE is derived by first estimating the baseline variance function for each of the compared experimental conditions, say U and V. For example, when duplicated arrays (U_1, U_2) are used for condition U, the variance of M (expressed as U_1-U_2) on each percentile range of A (expressed as U_1+U_2) is evaluated. When there are more than duplicates, all pairwise comparisons are pooled together for such estimation. A non-parametric local regression curve is then fit to the variance estimates on the percentile subintervals (refer to Figure 7.18 as an example). The baseline variance function for condition V is similarly derived, and the LPE test statistic for comparison of median log-intensities between the two samples is

$$z = (Med_1 - Med_2)/(s_{LPE})$$

where

$$s^2_{LPE} = 1.57(s_1{}^2(Med_1)/2n_1 + s_2{}^2(Med_2)/n_2)$$

n_1 and n_2 = numbers of replicates for the samples compared.

$s_i{}^2(Med_i)$, $i = 1, 2$ is the error estimate from the i-th LPE

baseline-error distribution at each median Med_i. For more details, see Lee and O'Connell (2003).

Note that the LPE statistic (based on medians) is robust to outliers if there are three or more replicates.

Raw P-Values

Running any of the above statistical procedures produces *raw* p-values, the p-values associated with the individual statistical tests. To make confident statements about differential expression for the entire

experiment, you need to compute *adjusted* p-values which control the family-wise error rate or false discovery rate. See the section Controlling The False Positive Rate for more details.

CONTROLLING TYPE I ERROR RATES

When testing for differential expression across many genes simultaneously, numerous genes may be identified as significantly differentially expressed by chance alone even if there is no real differential expression. For example, if you test 10,000 genes for differential expression at a significance level of 0.05, you can expect to misidentify about 500 genes as significant, even when there is no real difference in gene expression. Multiple testing corrections adjust the individual p-values to account for the inflated false positive rate due to multiple testing.

Because there are typically many genes represented in a microarray experiment, managing the side effects of multiple statistical tests is important in differential expression testing. Consequently, a number of procedures have been implemented in S+ARRAYANALYZER for controlling family-wise error rate (FWER) and false discovery rate (FDR).

Table 7.1: *Errors in statistical testing.*

Truth	**Significant Test**	**Not Significant Test**
Differentially Expressed	`S`	`FN = False Negative Type II Error`
Not Differentially Expressed	`FP = False Positive Type I Error =` α	`NS`
Total	`S + FP`	`NS + FN`

Controlling The False Positive Rate

Suppose the significance level of the test procedure is A and the number of genes being tested is N. A procedure is said to control the family-wise error rate (FWER) if it adjusts the significance level so that the overall error rate is, at most, A. Without adjusting the significance level, there may be as many as N false positives. For arrays with many genes, the number of false positives, without correcting for multiple tests, can be quite large.

Consequently, a number of procedures have been implemented in S+ARRAYANALYZER for controlling FWER and FDR. The results of the procedures are summarized using *adjusted* p-values, which reflect for each gene the overall Type I error rate when genes with a smaller p-value are declared differentially expressed.

Notation

Define $p_{(i)}, i = 1, \ldots, N$

where

N = number of comparisons (genes tested), as the ordered p-values (from smallest to largest) resulting from the statistical tests

$t_{(i)}$ = ordered test statistics from largest to smallest

$i = 1, \ldots, N$

H_0 = Null hypothesis - no differential expression

Then the p-value adjustment procedures are defined below.

FWER Procedures

Bonferroni:The Bonferroni correction is

$$p^* = \min(p_{(i)}N, 1) \text{ for each i.}$$

All genes with adjusted p-values, p*, less than α are significant with an overall FWER of at most α. Note that the raw p-values have simply been multiplied by the number of comparisons.

Hochberg: The Hochberg (1988) *step-down* correction is

$$p_{(i)}^* = \min_{k= i \ldots N}\{\min((N-k+1)p_{(k)}, 1)\}$$

The procedure sequentially computes

$$p_{(1)}^* = \min \{Np_{(1)}, (N-1)p_{(2)} , \ldots , p_{(N)}, 1\}$$

$$p_{(2)}^* = \min \{(N-1)p_{(2)}, (N-2)p_{(3)}, \ldots , p_{(N)}, 1\}$$

. . .

$$p_{(N-1)}^* = \min \{2p_{(N-1)}, p_{(N)}, 1\}$$

$$p_{(N)}^* = \min\{p_{(N)}, 1\}$$

and stops at the first adjusted p-value, $p_{(k)}^*$, that exceeds α.

Holm:The Holm (1979) *step-down* correction is

$$p_{(i)}^* = \max_{k=1\ldots i}\{\min((N-k+1)p_{(k)}, 1)\}$$

The procedure sequentially computes

$$p_{(1)}^* = \max \{\min (Np_{(1)}, 1)\}$$

$$p_{(2)}^* = \max \{\min (Np_{(1)}, 1), \min ((N-1)p_{(2)}, 1)\}$$

. . .

and stops at the first adjusted p-value, $p_{(k)}^*$, that exceeds α.

Sidak SS: The Sidak *single-step* (SS) correction is

$$p^* = 1 - (1 - p)^N$$

All genes with adjusted p-value, p*, less than α are significant with an overall FWER of at most α.

Sidak SD: The Sidak *free step-down* (SD) correction is

$$p_i^* = 1 - (1 - p_{(i)})^{(N-i+1)}$$

All genes with adjusted p-value, p_i^*, less than α are significant with an overall FWER of at most α.

minP: The Westfall and Young (1993) *minP step-down* procedure is computed as

$$p_{(i)}^* = \max_{k=1\ldots i}\{\Pr(\min_{j \text{ in } \{k\ldots N\}} Pj <= p_{(k)} | H_0)\}$$

For each $p_{(i)}$, $p_{(i)}^*$ is the *resampling-based* probability of obtaining a p-value no larger than $p_{(i)}$ from simulated probability distributions generated by the decreasing sets $\{p_{(1)},\ldots, p_{(N)}\}$, $\{p_{(2)},\ldots, p_{(N)}\}$,..., $\{p_{(i)},\ldots, p_{(N)}\}$.

maxT: The Westfall and Young (1993) *maxT step-down* procedure is computed as

$$p_{(i)}^* = \max_{k=1...i}\{\Pr(\max_{j \text{ in } \{k...N\}}|T_j| >= |t_{(k)}| \mid H_0)\}$$

For each $p_{(i)}$, $p_{(i)}^*$ is the *resampling-based* probability of obtaining a test statistic at least as large as $t_{(i)}$ from simulated distributions of the test statistics generated by the decreasing sets $\{t_{(1)},..., t_{(N)}\}$, $\{t_{(2)},..., t_{(N)}\}$,..., $\{t_{(i)},..., t_{(N)}\}$.

The **minP** and **maxT** procedures are only available for the permute versions of the test procedures (t-permute, t.equalvar-permute, wilcoxon-permute). Furthermore, the permute versions of the test statistics only have access to these two procedures for p-value adjustment.

The other adjustment procedures are implemented for all the non-permutation testing procedures described in the section Statistical Tests. The results of the procedures are summarized using adjusted p-values, which reflect for each gene the overall experiment Type I error rate when genes with a smaller p-value are declared differentially expressed. Adjusted p-values may be obtained either from the nominal distribution of the test statistics or by permutation.

FDR Procedures

The false discovery rate (FDR) is defined as the proportion of genes expected to be identified by chance relative to the total number of genes with significant tests of difference. That is, FDR = FP/(IS + FP) in Table 7.1. Controlling the FDR has the advantage of maintaining a small number of false positives amongst only those tests which are significant.

BH: The Benjamini and Hochberg procedure computes

$$p_{(i)}^* = \min_{k=i...N}\{\min((N/k)p_{(k)}, 1)\}$$

Any $p_{(i)}^* < \alpha$ is significant with an overall FDR for the experiment not greater than α. This procedure provides a good balance between discovery of significant genes and protection against false positives, since occurrence of the latter is held to a small proportion of the significant gene list.

BY: The Benjamini and Yekutieli procedure computes

$$p_{(i)}^* = \min_{k=i...N}\{\min(N\text{sum}_j((1/j)/k)p_{(k)}, 1)\}$$

Any $p_{(i)}^* < \alpha$ is significant with an overall FDR for the experiment not greater than α.

GUI FOR TWO SAMPLE TESTING

Two Sample Dialog Input

The dialog for **Two Sample Tests** is displayed in Figure 7.1. Open the dialog from the main S-PLUS menu by clicking **ArrayAnalyzer ▶ Differential Expression Analysis ▶ Two Sample Tests**.

The dialog is arranged in four main groups:

- **Data**
- **Options**
- **Graph Options**
- **Output**

Data

The **Data** group allows you to select the expression object for testing. You start by selecting the data type in **Show Data of Type** as one of Affymetrix or cDNA and then selecting a data object (an expression object created by importing, expression summarization (for Affy CEL), and normalization) from the **Data** drop-down list box.

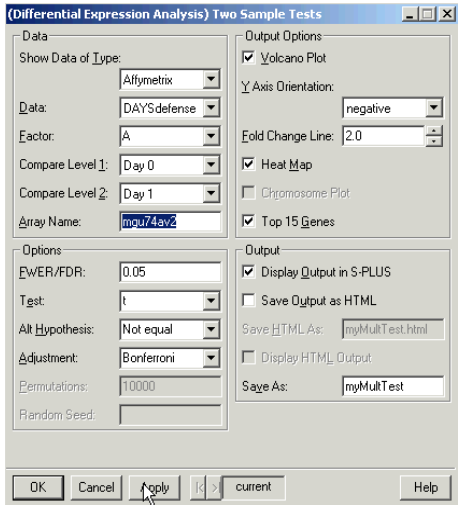


Figure 7.1: *The **Two Sample Test** dialog.*

Once a data object is selected, the chip name is filled in the **Chip Name** field. For custom 2-color cDNA or non-Affymetrix oligonucleotide chips, the chip name may be **<undetermined>**.

Options

The **Options** group, displayed in Figure 7.2, allows you to specify various options for the statistical tests:

1. Select the statistical test (default is **Welch's t-test**)
2. Specify the alternative (default is **Not equal**)
3. Input the FWER or FDR
4. Select the p-value adjustment procedure
5. Specify the number of iterations along with a random seed if you select one of the permutation methods.

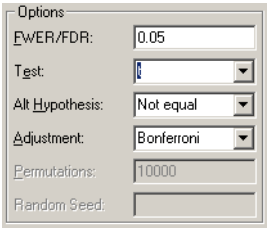


Figure 7.2: *The **Options** group of the **Two Sample Tests** dialog.*

Statistical Tests

The statistical tests and p-value adjustment procedures are all described in the section Statistical Tests and the section Controlling Type I Error Rates. The key words or phrases used to select one of these options match those used in the descriptive text.

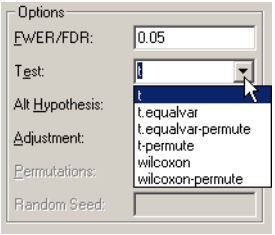


Figure 7.3: *Selecting a statistical test procedure in the **Options** group.*

FWER and FDR Control

The procedures for controlling the FWER and FDR are shown in the drop-down list of Figure 7.4. The procedures correspond to those described in the section Controlling The False Positive Rate and the section FDR Procedures. Both FWER and FDR procedures are

included in the drop-down list. For something other than the default Bonferroni correction with FWER = 0.05, select an adjustment procedure from the drop-down list and input the overall error rate in the **FWER** editable field.

Figure 7.4: *Procedures for controlling Type I Error rates.*

Note that the **minP** and **maxT** procedures are only available for the **permute** versions of the test statistics. When you use the permutation methods, you can specify the number of permutations used in the p-value estimation and provide a seed to the random number generator for repeatability of results in testing or validation studies.

Cautionary Note

The permutation and **minP** and **maxT** adjustment procedures should not be used in the context of few replicates; the results may be misleading. For more information, see the cautionary note in the section Within-Gene Two-Sample Comparisons of section Statistical Tests.

Output Options The **Output Options** group is a list of check boxes for selecting which graphs you want as output.

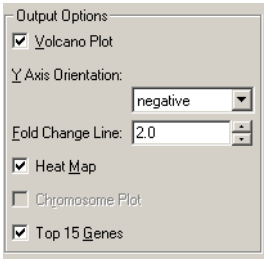


Figure 7.5: *The **Graph Options** group in the **Two Sample Test** dialog.*

Each of these options is described in detail in the section Differential Expression Analysis Plots.

Output

The **Output** group controls where the graphs are displayed and the gene list table is saved after the testing step is complete.

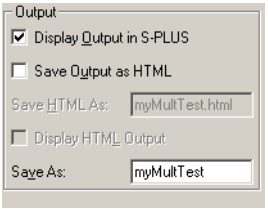


Figure 7.6: *The **Output** group of the **Two Sample Test** dialog.*

- **Display Output in S-PLUS**: Displays the selected graphics in an S-PLUS graphic device.
- **Save Output as HTML**: Saves the S-PLUS Graphlet with selected graphs and the significant gene list to HTML files to view later.
- **Display HTML Output**: View the S-PLUS Graphlet with selected graphs in a browser. The displayed Graphlet has a hyperlink to the significant genes table. Points on the Graphlet and entries in the significant gene list are automatically hyperlinked to LocusLink and UniGene annotation databases.
- **Save Summary As**: Name used for saving the S-PLUS data frame containing the *complete* gene list including test statistics and p-values. The default name is **myMultTest**.

Output Files

There are two output files generated when you select **Save Output as HTML**:

1. A significant genes summary table. The name of the this table is generated by taking the name supplied in the **Save Summary As** field and then adding **myMultTest.html**. The default supplied name is **myMultTest** so the default output table name is **myMultTest.html**.
2. An S-PLUS Graphlet with selected graphs. The name of the output files is generated by the name supplied in the **Save Summary As** field and then adding **myMultTest.html**. The default Graphlet name is **myMultTest.html**.

Location of Output Files

The location of these output files is determined by your S-PLUS working directory. To determine your working directory, just type

```
> getenv("S_WORK")
"D:\\arrayanalyzer\\users\\lenk\\test"
```

The location of "dumped" files in general is the default S-PLUS working directory. If you specify no project folder when you start S-PLUS, your **cmd** directory is the default working directory:

```
> getenv("S_WORK")
"D:\\Program Files\\Insightful\\splus62\\cmd"
```

You should see two HTML files in your working directory when S-PLUS has finished generating the output: one for the summary table and the another for the Graphlet.

GUI FOR LPE TESTING

LPE Testing Dialog Input

The dialog for LPE testing is displayed in Figure 7.7. Open the dialog from the main S-PLUS menu by clicking **ArrayAnalyzer ▶ Differential Expression Analysis ▶ LPE Test**.

The dialog is arranged in five main groups:

1. **Data**
2. **Options**
3. **Variance Estimator**
4. **Graph Options**
5. **Output**

Data

The **Data** group allows you to select the expression object for testing. You start by selecting the data type (**Show Data Type**) as one of Affymetrix or cDNA and then selecting a data object (an expression object created by importing, expression summarization (for Affy CEL), and normalization) from the **Data** drop-down list box.

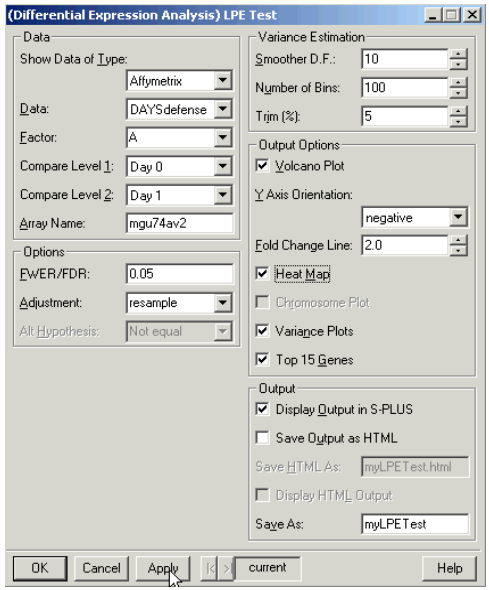


Figure 7.7: *The **Local Pooled Error Test** or LPE test dialog.*

Once a data object is selected the chip name is filled in the **Chip Name** field. For custom 2-color cDNA or non-Affymetrix oligonucleotide chips, the chip name may be **<undetermined>**.

Options

The **Options** group contains the procedures for controlling the FWER and FDR, as shown in the drop-down list in Figure 7.8. The procedures correspond to those described in section Controlling The False Positive Rate and section FDR Procedures. Both FWER and FDR procedures are included in the drop-down list. Select one and specify the family-wise error rate (for either an FWER or FDR procedure in the **FWER** editable field.

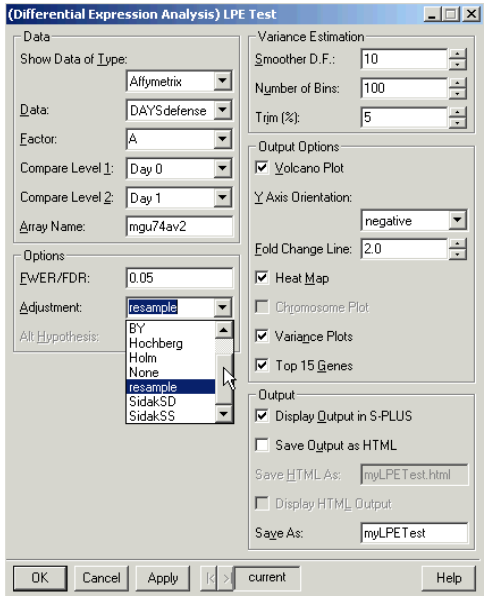


Figure 7.8: *Setting the p-value adjustment procedure for controlling the FWER.*

Variance Estimation

The **Variance Estimation** group in the upper right-hand corner of the dialog controls optional settings for the LPE estimator. The options are:

Smoother D.F.: The degrees of freedom used by the spline smoother to estimate the baseline variance function for each group. Default is 10.

Number of Bins: Number of bins to compute variance estimates. These variance estimates along with an associated average expression intensity is the data used by the loess smoother to estimate the baseline variance function.

Trim (%): Percent of pooled variances to trim from the low end of expression intensity prior to running the loess smoother.

Output Options

The **Output Options** group is a list of check boxes for selecting which graphs you want as output. The options are described in detail

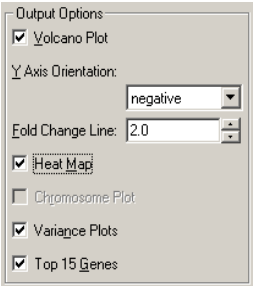


Figure 7.9: *The **Graph Options** group in the LPE test dialog.*

in the section Differential Expression Analysis Plots.

Output

The **Output** group controls where the graphs are displayed and the gene list table is saved after the testing step is complete.

- **Display Output in S-PLUS**: Displays the selected graphics in an S-PLUS graphic device.
- **Save Output as HTML**: Saves the S-PLUS graphlet with selected graphs and the significant gene list to HTML files to view later.
- **Display HTML Output**: View the S-PLUS Graphlet with selected graphs in a browser. The displayed Graphlet has a hyperlink to the significant genes table. Points on the Graphlet and entries in the significant gene list are automatically hyperlinked to LocusLink and UniGene annotation databases.

- **Save Summary As**: Name used for saving the S-PLUS data frame containing the *complete* gene list including test statistics and p-values. The default name is **myLPETest**.

Output Files

There are two output files generated when you select **Save Output as HTML**:

1. A significant genes summary table. The name of the this table is generated by taking the name supplied in the **Save Summary As** field and then adding **myLPETest.html**. The default supplied name is **myLPETest**, so the default output table name is **myLPETest.html**.
2. An S-PLUS Graphlet with selected graphs. The name of the output files is generated by the name supplied in the **Save Summary As** field and then adding **myLPETest.html**. The default Graphlet name is **myLPETest.html**.

Location of Output Files

The location of these output files is determined by your S-PLUS working directory. To determine your working directory, just type

```
> getenv("S_WORK")
"D:\\arrayanalyzer\\users\\lenk\\test"
```

The location of "dumped" files in general is the default S-PLUS working directory. If you specify no project folder when you start S-PLUS, your **cmd** directory is the default working directory:

```
> getenv("S_WORK")
"D:\\Program Files\\Insightful\\splus62\\cmd"
```

You should see two HTML files in your working directory when S-PLUS has finished generating the output: one for the summary table and the another for the Graphlet.

GUI FOR ANOVA TESTING

ANOVA Testing Dialog Input

The dialog for ANOVA testing is displayed in Figure 9.10. Open the dialog from the main S-PLUS menu by clicking **ArrayAnalyzer ► Differential Expression Analysis ► ANOVA**.

The dialog is arranged in five main groups:

1. **Data**
2. **Contrasts**
3. **Options**
4. **Output Options**
5. **Output**

Data

The **Data** group allows you to select the expression object for testing. You start by selecting the data type (**Show Data Type**) as one of Affymetrix or cDNA and then selecting a data object (created through import, probe-level summary (for Affy CEL), and normalization) from the **Data** drop-down list box.

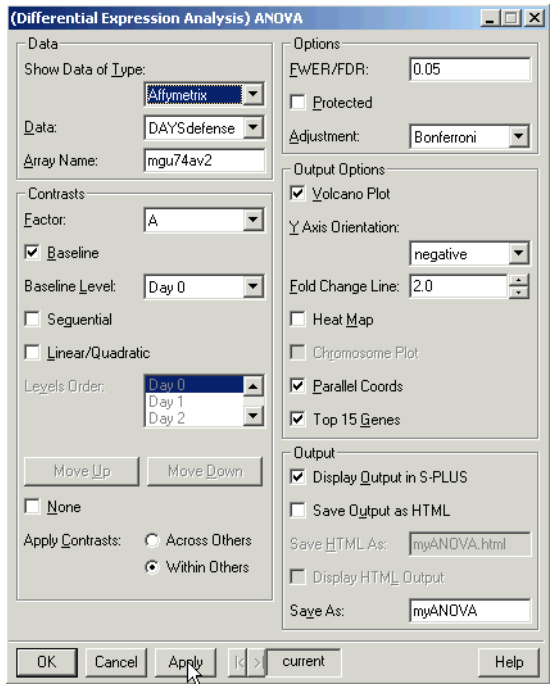


Figure 7.10: *The **ANOVA** test dialog.*

Once a data object is selected the chip name is filled in the **Chip Name** field. For custom 2-channel or non-Affymetrix oligonucleotide chips, the chip name may be **<undetermined>**.

Contrasts

The **Contrasts** group contains settings for selecting the types of comparisons done between the levels of the factors. The selected contrasts are 'orthogonal' and may be considered as independent sources of information extracted from the experimental data. The contrast Factor is chosen first. There are 3 choices for the contrasts on this chosen factor: **Baseline**, **Sequential**, **Linear/Quadratic** and **None**.

The **Baseline** contrast setting compares the levels of the chosen **Factor** to the **Baseline Level** selected. This is commonly used in a time course or concentration series experiment in which levels of a **Factor** representing the different time points are contrasted against the **Baseline Level** as the initial time point (or any other reference time point). For example, if a factor time had levels (t0, t1, t2, t3) and t0 was chosen as baseline level, the contrasts would be t1-t0, t2-t0, t3-t0 i.e. the differences between baseline and each subsequent time. The contrast analysis identifies genes that are differentially expressed at each particular time point compared to baseline. These differentially expressed genes for each contrast provide visibility into the biological sequence of events across the time course.

The **Sequential** setting compares levels of the chosen factor in sequential ordered pairs. For example, if a factor time had levels (t0, t1, t2, t3), the contrasts would be t1-t0, t2-t1, t3-t2. These contrasts also provide systematic view of differential expression over the time course or factor levels. The choice of Sequential vs. Baseline may depend on whether a true baseline condition was run and/or the interpretive goals of the experiment.

The **Linear/Quadratic** setting fits linear and quadratic contrasts across the (ordered) levels of the chosen factor. This analysis identifies genes that exhibit (a) a significant linear trend in either the positive (up) or negative (down) direction; and (b) a significant quadratic trend in either the positive (down in the middle) or negative (up in the middle) direction.

Contrasts may be applied either **Within** or **Across** another factor. For example if there are two sexes (M, F) tested at each of the 4 time points mentioned above i.e. (t0, t1, t2, t3), testing the time contrasts vs. baseline **Within** the Sex factor would result in the comparisons (t1-t0, t2-t0, t3-t0) for each of the 2 sexes. Testing the time contrasts vs. baseline **Across** the Sex factor would result in just one set of comparisons (t1-t0, t2-t0, t3-t0) averaged across the 2 sexes. Similarly if Sex was chosen as the contrast factor, testing the Sex contrast Within the Time factor would result in the comparisons (M-F at t0, M-F at t1, M-F at t2, M-F at t3); and testing the Sex contrast Across the Time factor would result in a single sex comparison M-F.

Options

The **Options** group contains the procedures for controlling the FWER and FDR, as shown in the drop-down list in Figure 9.11. The procedures correspond to those described in section Controlling The

False Positive Rate and section FDR Procedures. Both FWER and FDR procedures are included in the drop-down list. Select one and specify the family-wise error rate (for either an FWER or FDR procedure) in the **FWER** editable field.

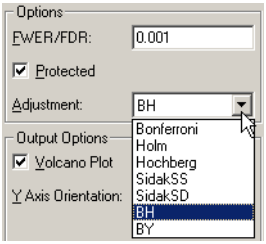


Figure 7.11: *Setting the p-value adjustment procedure for controlling the FWER.*

Output Options

The **Output Options** group is a list of check boxes and a drop-down box for selecting which graphs you want as output. The options are

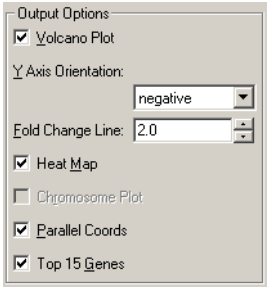


Figure 7.12: *The **Graph Options** group in the LPE test dialog.*

described in detail in the section Differential Expression Analysis Plots.

Output

The **Output** group controls where the graphs are displayed and the gene list table is saved after the testing step is complete.

- **Display Output in S-PLUS**: Displays the selected graphics in an S-PLUS graphic device.
- **Save Output as HTML**: Saves the S-PLUS graphlet with selected graphs and the significant gene list to HTML files to view later.
- **Display HTML Output**: View the S-PLUS Graphlet with selected graphs in a browser. The displayed Graphlet has a hyperlink to the significant genes table. Points on the Graphlet and entries in the significant gene list are automatically hyperlinked to LocusLink and UniGene annotation databases.
- **Save Summary As**: Name used for saving the S-PLUS data frame containing the *complete* gene list including test statistics and p-values. The default name is **myANOVA**.

Output Files

There are two output files generated when you select **Save Output as HTML**:

1. A significant genes summary table. The name of the this table is generated by taking the name supplied in the **Save Summary As** field and then adding **myANOVA.html**. The default supplied name is **myANOVA** so the default output table name is **myANOVAhtml**.
2. An S-PLUS Graphlet with selected graphs. The name of the output files is generated by the name supplied in the **Save Summary As** field and then adding **myANOVA.html**. The default Graphlet name is **myANOVA.html**.

Location of Output Files

The location of these output files is determined by your S-PLUS working directory. To determine your working directory, just type

```
> getenv("S_WORK")
"D:\\arrayanalyzer\\users\\lenk\\test"
```

The location of “dumped” files in general is the default S-PLUS working directory. If you specify no project folder when you start S-PLUS, your **cmd** directory is the default working directory:

```
> getenv("S_WORK")
```

```
"D:\\Program Files\\Insightful\\splus62\\cmd"
```

You should see two HTML files in your working directory when S-PLUS has finished generating the output: one for the summary table an

DIFFERENTIAL EXPRESSION ANALYSIS PLOTS

Common Plots

The differential expression summary plots are designed to give you easy access to annotation data in public databases. Two of the plots, the volcano plot and the heat map, have embedded hyperlinks so you can click on a point and bring up annotation from NCBI databases. There are three plots common to both testing dialogs:

- Volcano plot
- Heat map
- Chromosome plot

Each of the dialogs optionally produces one additional plot. The **Two Sample Tests** dialog produces a Q-Q Normal Probability plot of the test statistics and the LPE Test dialog produces a Variance plot displaying a graph of the baseline variance estimates as a function of the average expression intensity for each experimental condition. The ANOVA Test produces Principal Components Plots, Volcano Plots and Summary Table for each contrast selected as well as a heat map.

Each of these types of plots is discussed in the following sections.

Volcano Plot

A *volcano* plot displays the logarithm of adjusted p-value versus average fold change. The vertical lines indicate average fold change values of plus or minus two, and the horizontal line indicates a significant *adjusted* p-value. Points located in the lower, outer sextants (using this graphing orientation) are those with large absolute fold change and small (significant) p-value. Each of those points is active so you can click an individual point to access annotation from LocusLink or UniGene databases. Volcano Plots may now be plotted in positive (see below) or negative (classic volcano style) orientation

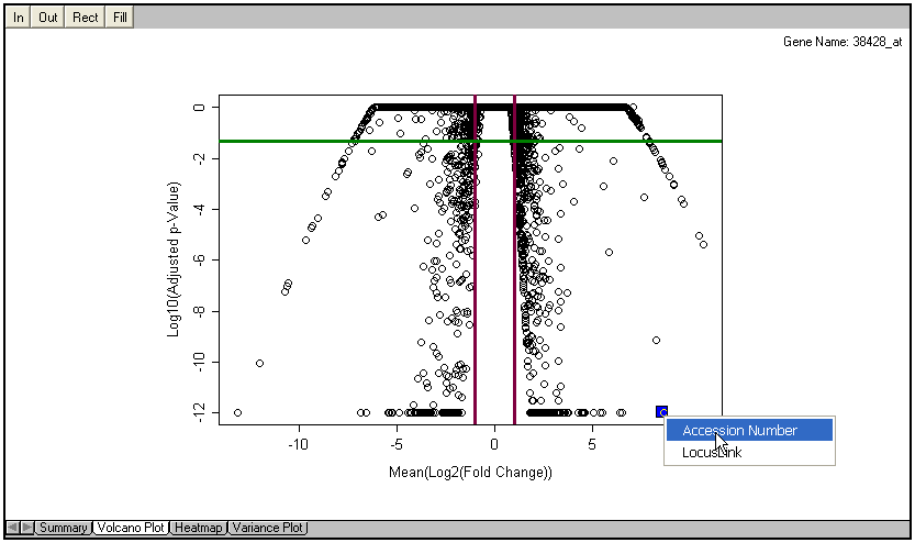


Figure 7.13: *A volcano plot shows the logarithm of the adjusted p-value vs. average fold change. This is displayed using an S-PLUS Java graphics device. It may also be displayed in the browser.*

for all the forms of differential expression testing.

An example of the The volcano plot complete with hyperlinks can be sent to an HTML file for later viewing. It can also be sent to an S-PLUS graphics window. Figure 7.13 shows a typical volcano plot with the interactive menu generated by clicking a point in the differential expression region of the plot.

When the plot is viewed in an S-PLUS graphics window the active points are not hyper-linked to the annotation databases. However hovering the mouse over active points shows the gene name ID in the upper right corner of the graphic as it does for the HTML display., as shown in Figure 7.14.

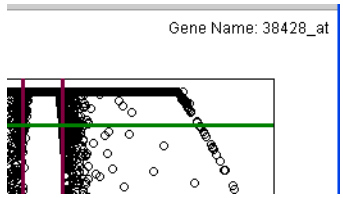


Figure 7.14: *Finding the gene name on the graph for differentially expressed genes.*

Heat map

A *heat map* plot shows a 2-D image plot of the 300 genes with lowest p-values (by default) along the vertical axis versus the experimental conditions on the horizontal axis. See Figure 7.15. This graph is also hyperlinked to public annotation databases and displays the gene identifier in the upper right corner of the plot. Left-clicking in a colored rectangle exposes the menu for making an annotation database choice.

The left and top margins of the graph contain a dendrogram resulting from applying hierarchical clustering to the expression intensity values and treatment conditions, respectively.

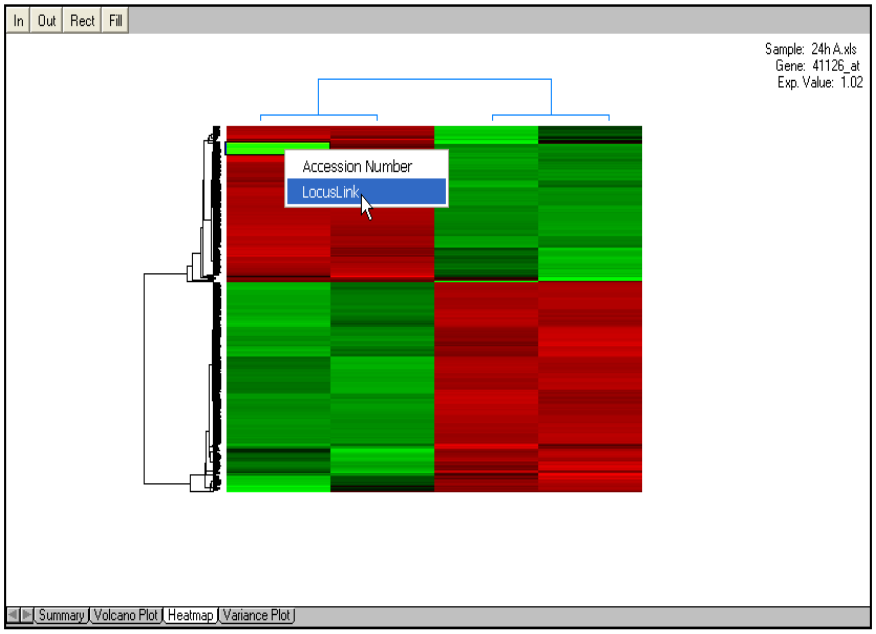


Figure 7.15: *Heat map plot for differentially expressed genes. This graphlet may be displayed through a Web browser or an S-PLUS Java graphics device. Pixels colored red signify positive expression values; those colored green signify negative expression values. The brighter the color, the larger the intensity in absolute value.*

Chromosome Plot A *chromosome* plot displays the *human* genome for Affymetrix's HG-U95A chip. Differential expression is marked for *up*-regulation and *down*-regulation for each gene represented on the chip. The top 10 differentially expressed genes are highlighted with color (orange) to indicate their location on the chromosome. Hovering the mouse over one of the colored (active) points displays the gene ID in the upper right-hand corner of the graph, as shown in Figure 7.16.

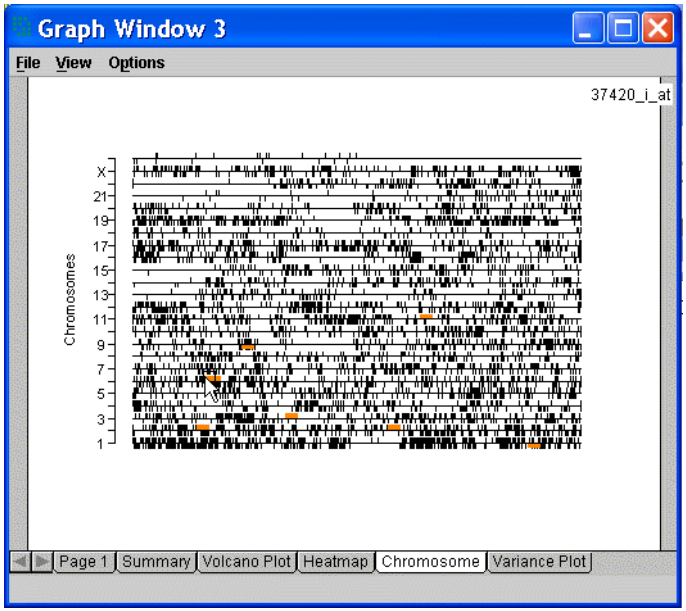


Figure 7.16: *Chromosome plot of the human genome for Affymetrix's HG-U95A chip with the 10 most differentially expressed genes displayed in color. Hovering the mouse over the colored spots displays the gene ID in the upper right corner.*

Two Sample Tests Specific Plots

For the **Two Sample Tests** dialog, in addition to the volcano, heat map and chromosome plots, you may also generate a Q-Q Normal Probability plot of the test statistics. This plot provides a visual assessment of the distribution of the test statistics relative to the standard normal distribution, as shown in Figure 7.17.

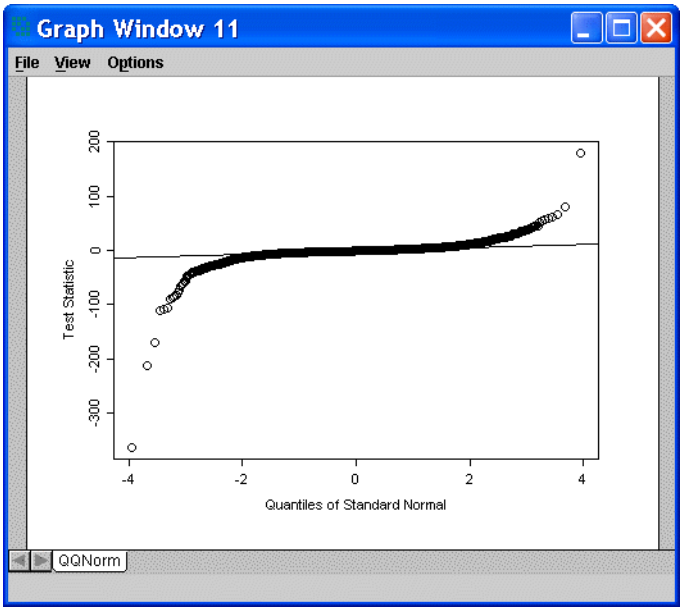


Figure 7.17: *Q-Q Normal Probability plots of the test statistics generated by the* ***Two Sample Tests*** *dialog.*

LPE Specific Plots For the **LPE Test** dialog, in addition to the volcano, heat map and chromosome plots, you may also generate plots of the local pooled error variance versus the overall intensity within experimental conditions. Two plots are produced, one for each experimental condition, as shown in Figure 7.18.

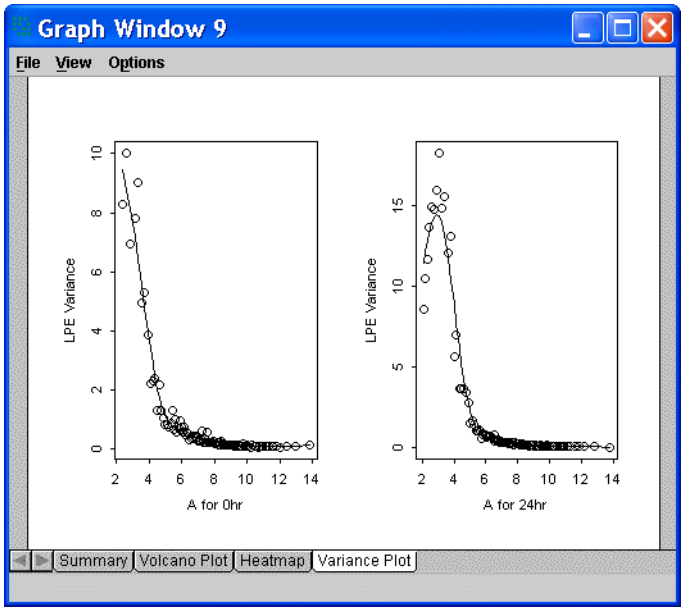


Figure 7.18: *Plots of the local pooled error variance within treatments versus the overall intensity within treatments.*

ANOVA Specific Plots

For the **ANOVA Test** dialog, in addition to the volcano for each contrast and heat map, you may also generate parallel coordinates plots for each contrast. A multiple plots are produced, one for each contrast, a single contrast is shown in Figure 7.18.

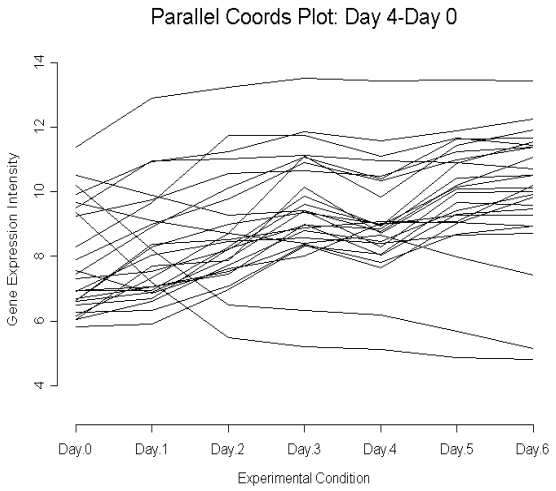


Figure 7.19: *Parallel Coordinates Plot for a single ANOVA contrast.*

DIFFERENTIAL EXPRESSION SUMMARY TABLE OUTPUT

In addition to graphical output, the differential expression testing functions generate summary tables ordering the gene list from most to least differentially expressed. The graphical (Graphlet) output provides a list of the top 15 genes, but you get the complete gene list as well.

Top 15 List

The 15 genes with the lowest p-values are displayed in the Graphlet with the volcano plot, heat map and other plots.

Summary Output for ANOVA Test-Day 1-Day 0
with BH Adjustment - Top 15 Genes

	Test Stat	Raw p-Val	Adj. p-Val
neutrophilic granule protein	> 10	0	0
immunoglobulin heavy chain 6 (heav	> 10	0	0
T-cell receptor gamma, variable 2	< -10	0	0
interferon activated gene 205	> 10	< 0.001	< 0.001
complement component 3	> 10	< 0.001	< 0.001
CD24a antigen	> 10	< 0.001	< 0.001
T-cell receptor gamma, variable 4	< -10	< 0.001	< 0.001
integrin beta 2	> 10	< 0.001	< 0.001
complement component 4 (within H-2	> 10	< 0.001	< 0.001
lymphocyte antigen 6 complex, locu	> 10	< 0.001	< 0.001
cathelicidin antimicrobial peptide	> 10	< 0.001	< 0.001
neutrophil cytosolic factor 1	> 10	< 0.001	< 0.001
immunoglobulin heavy chain 6 (heav	> 10	< 0.001	< 0.001
proteoglycan 2, bone marrow	> 10	< 0.001	< 0.001
complement component 1, q subcompo	-9.65	< 0.001	< 0.001

Figure 7.20: *15 most differentially expressed genes.*

Complete Gene List

The *complete* gene list is saved in an S-PLUS object. For more details see section Output in section Two Sample Dialog Input and in section LPE Testing Dialog Input in this chapter. You can access the gene list in three different ways:

1. The **Data ► Select Data** menu item on the main S-PLUS menu bar.
2. Through the S-PLUS **Object Explorer**.

3. The Command line.

From The Data Menu Item

Open the **Select Data** dialog by selecting **Data ▶ Select Data** from the main S-PLUS menu bar and select the test summary objects from the **Existing Data Name** drop-down list.

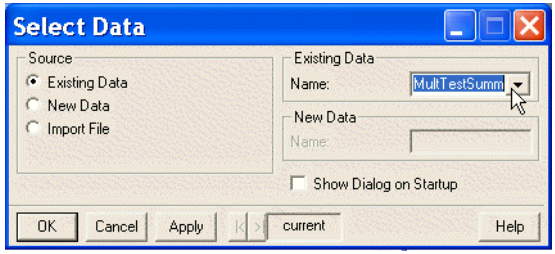


Figure 7.21: *Selecting the complete gene list from the Select Data dialog.*

Clicking **OK** opens a data sheet containing the summary information.

MultTestSumm (Read Only)

		1	2	3	4	5	6	7	8
		gName	mean.0hr	mean.24hr	foldChange	testStat	rawp	adjp	signif.p
1	35704_at	35704_at	9.24	0.54	8.70	188.04	0.00	0.11	T
2	37023_at	37023_at	8.74	0.54	8.20	174.87	0.00	0.11	T
3	33532_at	33532_at	7.78	10.90	-3.13	-3621.54	0.00	0.11	T
4	37712_g_at	37712_g_at	8.47	0.54	7.93	163.33	0.00	0.11	T
5	31979_at	31979_at	7.21	0.54	6.67	142.71	0.00	0.11	T
6	1837_at	1837_at	7.44	0.54	6.90	164.14	0.00	0.11	T
7	41848_f_at	41848_f_at	8.59	0.54	8.05	150.70	0.00	0.12	T
8	1984_s_at	1984_s_at	8.43	0.54	7.89	199.13	0.00	0.12	T
9	41231_f_at	41231_f_at	12.93	13.80	-0.87	-115.31	0.00	0.13	T
10	36250_at	36250_at	8.62	9.07	-0.45	-109.52	0.00	0.13	T
11	37777_at	37777_at	4.75	0.54	4.21	98.81	0.00	0.13	T

Figure 7.22: *The first few rows of the gene list summary table generated by the **Two Sample Test** dialog.*

From The Object Explorer

Open the S-PLUS **Object Explorer** by clicking the **Object Explorer** tool bar button displayed in Figure 7.23.

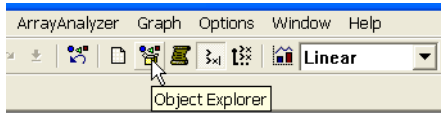


Figure 7.23: *S-PLUS Object Explorer tool bar button.*

Under the **Data** tree in the **Object Explorer**, double-click the summary object of your choice. See Figure 7.24.

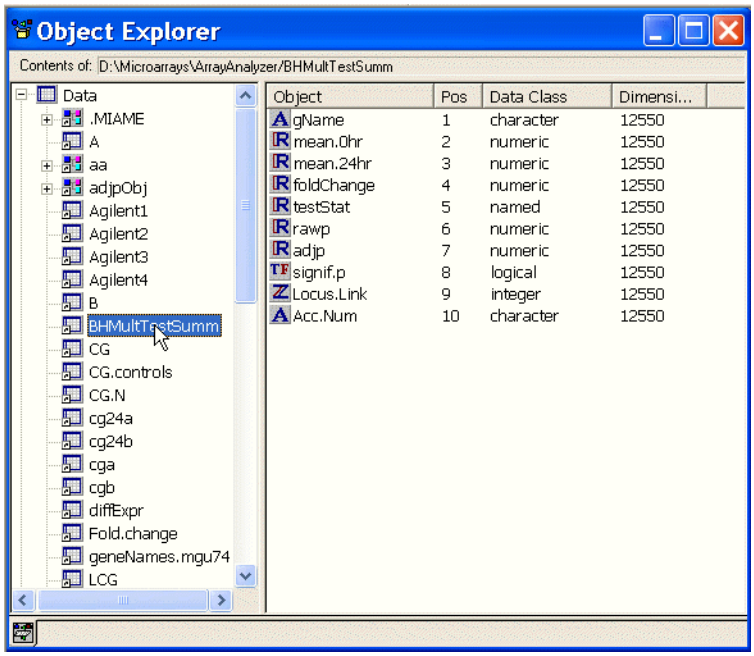


Figure 7.24: *The **Object Explorer** in S-PLUS allows you to browse your data files and open them in a grid for viewing.*

From the Command Line

To access the gene list from the Command line, you need the object name. The default output names for the **Two Sample Test**, **LPE Test and ANOVA** dialogs are **myMultTest**, **myLPETest and myANOVA**, respectively. For an object with FDR set to 0.001 and Benjamini-Hochberg adjustment, the first 5 rows of one object (named **myLPETest**) looks as follows:

```
> slotNames(myLPETest)
[1] "allData"   "means"     "annoData"  "contrData" "testInfo"
"dataInfo"


> myLPETest@allData[1:10,]
GeneName GeneIndex foldChange Pvalue   testStat AdjPvalue Signif.p
 97763_at  97763_at       63  -4.208443     0 -44.932855       0
T
160169_at 160169_at       64   1.178213     0   8.542566       0
T
```

```
101918_at 101918_at      80   1.074805     0   9.563393        0
T
 97528_at  97528_at     113   1.639526     0  11.371125        0
T
101475_at 101475_at     284   1.186268     0   8.712748        0
T
102744_at 102744_at     383   4.552544     0  33.561404        0
T
102745_at 102745_at     384   5.062317     0  39.989042        0
T
104093_at 104093_at     497  -4.614482     0 -48.593687        0
T
 98000_at  98000_at     505   2.239734     0  17.949190        0
T
101820_at 101820_at     514  -5.753799     0 -63.326020        0
T
```

REFERENCES

Benjamini, Y., Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society,* Series B 57: 289-300.

Benjamini, Y., Yekutieli, D. (2001). The control of the false discovery rate in multiple hypothesis testing under dependency. *Annals of Statistics* 29(4): 1165-1188.

Dudoit, S., Shaffer, J. P., and Boldrick, J. C. (2002). Multiple hypothesis testing in microarray experiments. *U.C. Berkeley Division of Biostatistics Working Paper Series*. Working Paper 110.

Dudoit, S., Yang, Y., Callow, M., and Speed, T. (2002). Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Statistica Sinica*, 12:111-139.

Efron, B., Tibshirani, R., Storey, J. D. and Tusher V. (2001). Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association* 96: 1151-1160.

Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*. Vol. 75: 800-802.

Hochberg, Y. and Tamhane, A. C. (1987). *Multiple Comparison Procedures*. New York: Wiley.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Statist*. Vol. 6: 65-70.

Hsu, J. C. (1996). *Multiple Comparisons: Theory and Methods*. London: Chapman and Hall.

Lee, J. K., and O'Connell, M. (2003). An S-PLUS library for the analysis of differential expression. In *The Analysis of Gene Expression Data: Methods and Software*. Edited by G. Parmigiani, E.S. Garrett, R.A. Irizarry and S.L. Zeger. Springer, New York.

Moore, D. S. and McCabe, G. P. (1999). *Introduction to the Practice of Statistics*, 3rd ed. New York: W. H. Freeman and Company.

Snedecor, G. W. and Cochran, W. G. (1980). *Statistical Methods*, 7th ed. Ames, Iowa: Iowa State University Press.

Storey J. D. (2002). A direct approach to false discovery rates. *Journal of the Royal Statistical Society*, Series B 64: 479-498.

Westfall, P. H. and Young, S. S. *Resampling-based multiple testing: Examples and methods for p-value adjustment.* John Wiley & Sons, 1993.

CLUSTER ANALYSIS 8

Introduction **302**

Hierarchical Methods **304**
- Agglomerative Nesting 304
- Model-Based Hierarchical Clustering 305
- Understanding The Results of Hierarchical Clustering 306

Partitioning Methods **308**
- K-Means Clustering 308
- Partitioning Around Medoids 308

Examples from the GUI **310**
- Basic Dialog Description 310
- Clustering Affymetrix Summary Data 311
- Clustering Two-Channel Expression Data 316
- Clustering Significant Genes From An ANOVA 317

Examples from the Command Line **320**

References **329**

INTRODUCTION

Cluster analysis is the searching for groups (clusters) in the data, in such a way that objects belonging to the same cluster resemble each other, whereas objects in different clusters are dissimilar. In two or three dimensions, clusters can be visualized. With more than three dimensions, we need some kind of analytical assistance. Generally speaking, clustering algorithms fall into two categories:

1. **Hierarchical Algorithm**: A hierarchical algorithm describes a method yielding an entire hierarchy of clustering for the given data set. Agglomerative methods start with the situation where each object in the data set forms its own little cluster, and then successively merges clusters until only one large cluster remains which is the whole data set. Divisive methods start by considering the whole data set as one cluster, and then splits up clusters until each object is separate.
2. **Partitioning Algorithms**: A partitioning algorithm describes a method that divides the data set into k clusters, where the integer k needs to specified. Typically, you run the algorithm for a range of k-values. For each k, the algorithm carries out the clustering and also yields a "quality index," which allows you to select the "best" value of k afterwards. Algorithms of this type include k-means and partitioning around medoids.

Clustering approaches have been widely applied to the analysis of gene expression data (Eisen et al., 1998; Scherf et al., 2000). In particular, the method of visualizing gene expression data based on cluster order, or cluster image map analysis using *hierarchical clustering*, has been found to be an efficient approach for summarizing thousands of gene expression values and assisting in the identification of interesting gene expression patterns. *Partitioning clustering* methods such as K-means are used to identify candidate subgroups in experiments involving multiple samples and/or experimental conditions. Both hierarchical and partitioning clustering have been used, for example, in the identification of novel sub-types of cancers.

Additional gene information is also extremely useful for discovering meaningful clustering patterns. Prior to clustering, it is also recommended to identify genes based on their statistical significance in differential expressions and to confirm consistent expression patterns within replicates (e.g. Ross et al., 2000).

S+ARRAYANALYZER provides both types of algorithms via it's GUI. Also, S-PLUS has additional algorithms belonging to both categories. You can access them either through the S-PLUS GUI (see **Statistics ▶ Cluster Analysis** from the main S-PLUS menu bar) or through the command line interface. A good general reference to clustering methods is Kaufman and Rousseeuw (1990).

HIERARCHICAL METHODS

Partitioning algorithms are based on specifying an initial number of groups, and iteratively reallocating observations between groups until some equilibrium is attained. In contrast, hierarchical algorithms proceed by combining or dividing existing groups, producing a hierarchical structure displaying the order in which groups are merged or divided.

Agglomerative methods start with each observation in a separate group, and proceed until all observations are in a single group. Divisive methods start with all observations in a single group and proceed until each observation is in a separate group.

Agglomerative Nesting

The basic hierarchical agglomeration algorithm starts with each object in a group of its own. At each iteration it merges two groups to form a new group; the merger chosen is the one that leads to the smallest increase in the sum of within-group sums of squares. The number of iterations is equal to the number of objects minus one, and at the end all the objects are together in a single group. This is known variously as Ward's method, the sum of squares method, or the trace method.

Linkage Methods

The hierarchical agglomeration algorithm can be used with criteria other than the sum of squares criterion. For example, in the *connected* (also known as the single linkage or nearest neighbor) method, the distance between two groups is defined to be the smallest distance between any two members from different groups, and at each iteration the two closest groups are merged. The *compact* method (also known as the complete linkage or farthest neighbor method) is similar except that the distance between any two groups is defined to be the largest distance between any two members from different groups. The *average* weighted linkage method uses the average of the distances between the objects in one group and the objects in the other group. These are all heuristic criteria.

Distance Metrics

The agglomerative hierarchical methods use several measures for defining *between-cluster* distance (or dissimilarity). These methods proceed by merging the two clusters with the smallest between-cluster variability, based on the chosen between-cluster dissimilarity measure, at each stage of the process.

Four distance metrics are available as options for heirarchical clustering through the S+ARRAYANALYZER GUI.

Euclidean: The square-root of the sum-of-squares of differences between points in the two groups.

Maximum: The maximum difference between points in the two groups.

Manhattan: The sum of absolute differences between points in the two groups.

Binary: The proportion of non-zeros that two vectors do not have in common (the number of occurrences of a zero and a one, or a one and a zero divided by the number of times at least one vector has a one).

Handling Missing Values

The distance methods handle missing values in different ways. Following is a brief description of each:

Euclidean: If ng is the number of arrays in which no missing values occur for the given genes, then the distance returned is sqrt(ncol(x)/ng) times the Euclidean distance between the two vectors of length ng shortened to exclude missing values.

Maximum: No special handling for missing values is necessary.

Manhattan: The rule is similar to the Euclidean metric, except that the coefficient is ncol(x)/ng.

Binary: The rule excludes columns in which either row has a missing value. If all values for a particular distance are excluded, the distance is labeled as missing.

Model-Based Hierarchical Clustering

Another approach to hierarchical clustering is model-based clustering. This method is based on the assumption that the data are generated by a mixture of underlying normal (or Gaussian)

probability distributions. It provides insight into the number of clusters, a quantity that is derived from a model selection process in its probability framework.

S-PLUS provides a model-based clustering algorithm for use at the command line or through the GUI. However, updated methods are now available for free download.

Understanding The Results of Hierarchical Clustering

Hierarchical methods have been widely used for the cluster analysis of microarray data. Yeung et al. (2001) discuss the benefits of model-based clustering for microarray analysis. Results from the hierarchical methods are typically represented with a dendrogram showing the hierarchy from all samples to individual samples or from all genes to individual genes. Genes with obviously non-significant expression values should be omitted from the clustering analyses. Genes included in the clustering analyses may be chosen using the statistical hypothesis tests for differential expression described in Chapter 7, Differential Expression Testing.

It is important to understand that hierarchical approaches do not directly provide any reliable measure of confidence for clustered expression patterns. A hierarchical clustering method heuristically reorganizes the genes based on its predefined association distance and allocation algorithm, which only aids in discerning co-expression patterns visually.

Therefore, a validation step is required for such hierarchical clustering discoveries before further inference can be drawn. For example, a bootstrapping method can be used for assessing reliability of clustering classifications of a fixed, known number of groups (Kerr and Churchill, 2001).

Hierarchical clustering results are typically summarized with a dendrogram, in which samples (or genes) are joined in a tree structure where the leaves/branches successively join samples (or genes) that are most similar. We note that this needs to be interpreted with care since hierarchical clustering imposes structure (whether it is there or not) and dendrograms then reflect that imposed structure. There are at least a couple of ways that the value of such analyses can be assessed.

The *cophenetic* distance between two observations i and j is defined to be the intergroup distance at which observations are first put into the same cluster. The extent to which cophenetic distances reflect the true distances relates to the usefulness of the dendrogram as a tool for visualization. This agreement can be assessed by the cophenetic correlation coefficient or the correlation between the true distances and the cophenetic distances.

The *silhouette* distance measures how well individual samples are classified into a discrete set of classes. This is a particularly relevant measure in assessing the value of a partitioning cluster analysis but can be applied to a hierarchical analysis by cutting the tree at some point and classifying samples into the groups defined by the cut. This is described further below.

PARTITIONING METHODS

The partitioning methods available through the S+ARRAYANALYZER GUI are K-Means and Partitioning Around Medoids.

K-Means Clustering

One of the most well-known partitioning methods is *k-means*. In the *k*-means algorithm the observations are classified as belonging to one of k groups. Group membership is determined by calculating the centroid for each group (the multidimensional version of the mean) and assigning each observation to the group with the closest centroid.

The k-means algorithm alternates between calculating the centroids based on the current group memberships, and reassigning observations to groups based on the new centroids. Centroids are calculated using least-squares, and observations are assigned to the closest centroid based on least-squares. This use of a least-squares criterion makes k-means less resistant to outliers than the medoid-based methods which will be discussed in later sections.

Partitioning Around Medoids

The partitioning around medoids (PAM) algorithm is similar to k-means but uses medoids rather than centroids.

The method PAM is fully described in Chapter 2 of Kaufman and Rousseeuw (1990). Compared to k-means clustering, PAM has the following features: (a) it accepts a dissimilarity matrix; (b) it is more robust because it minimizes a sum of dissimilarities instead of a sum of squared euclidean distances; (c) it provides novel graphical displays (silhouette plots and cluster plots).

Partitioning methods are appropriate when distinct sets of subpopulations are hypothesized. Results from using the partitioning methods are typically represented with cluster biplots and silhouette plots. Cluster biplots show the subpopulations separated in the first two principal component dimensions, whereas silhouette plots show how well individual samples are classified.

Silhouette Plots

In silhouette plots, for each object i (a sample or experimental condition typically), the silhouette value $s(i)$ is computed and then represented in the plot as a bar of $s(i)$. If A denotes the cluster to which object i belongs, we define

$$a(i) = \text{average dissimilarity of } i \text{ to all other objects of } A$$

Now consider any cluster C different from A and define

$$d(i, C) = \text{average dissimilarity of } i \text{ to all objects of } C$$

After computing $d(i, C)$ for all clusters C *not equal to* A, we take the smallest of them:

$$b(i) = \min_{C \neq A} d(i, C)$$

The cluster B that attains this minimum, namely $d(i, B) = b(i)$, is called the *neighbor* of object i. This is the second-best cluster for object i.

The value $s(i)$ can now be defined:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{8.1}$$

We see that $s(i)$ always lies between -1 and 1. The value $s(i)$ may be interpreted as follows:

$s(i) \approx 1 \Rightarrow$ object i is well classified

$s(i) \approx 0 \Rightarrow$ object i lies between two clusters

$s(i) \approx -1 \Rightarrow$ object i is badly classified

The silhouette of a cluster is a plot of the $s(i)$ ranked in decreasing order. The silhouette plot shows the silhouettes of all clusters next to each other so the *quality* of the clusters can be compared. The *average silhouette width* of a partitioning cluster analysis is the average of all the $s(i)$ from every cluster. This is a measure of quality, or goodness, of the cluster analysis.

One typically runs `pam` several times, using a different number of clusters within a specified range appropriate for the number of samples, and compares the resulting silhouette plots. One can then select the number of clusters yielding the highest average silhouette width. If the highest average silhouette width is small (e.g., below 0.2), one may conclude that no substantial structure has been found.

EXAMPLES FROM THE GUI

To access the clustering methods available through the S+ARRAYANALYZER GUI click **ArrayAnalyzer ▶ Cluster Analysis** from the main S-PLUS menu bar.

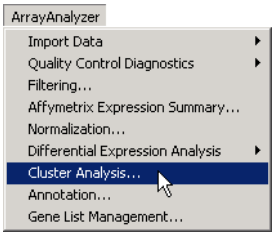


Figure 8.1: *Opening the Cluster Analysis dialog from the S-PLUS menu bar.*

Basic Dialog Description

The resulting dialog has two tabs

1. **General Options** for data and algorithm selection.
2. **Filtering Options** for subsetting the data before clustering.

The **General Options** tab is organized into five groups:

1. **Data**: to select the data set of interest
2. **Response Variable**: to select the clustering variable
3. **Hierarchical Methods**: to select hierarchical clustering and options.
4. **Partitioning Methods**: to select a partitioning methods and options.
5. **Output**: display options.

The Filtering Options tab has four groups:

1. **Contrast Filtering**: to filter on a contrast from an ANOVA test object.
2. **Expression Filtering**: to filter on expression values.
3. **Gene List Filtering**: to filter on an existing gene list
4. **Gene Sort Order Options**: options to sort filtering results.

.

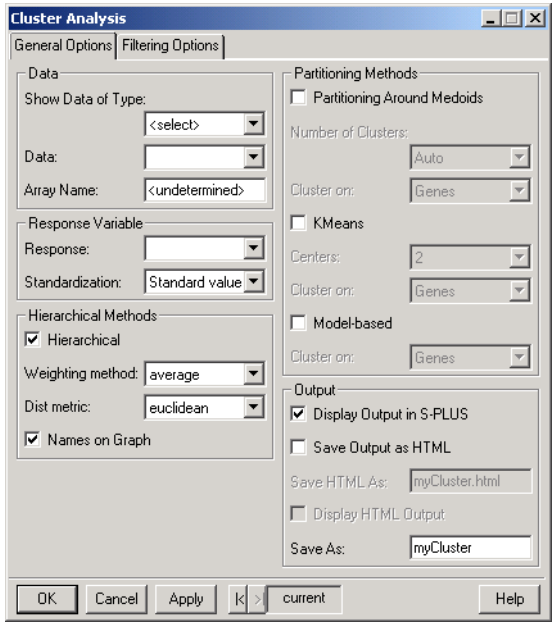


Figure 8.2: *The default Cluster Analysis dialog.*

Clustering Affymetrix Summary Data

To set up a clustering problem, start by selecting the data type from the **Show Data of Type** drop-down list. Notice that you can choose Affymetrix or two-channel expression objects *or* differential expression test objects.

Hierarchical

For our first example, we'll choose the swimming mice expression object created in Chapter 2, Examples: Affymetrix MAS Data. If you haven't worked through this example yet, go to Chapter 2 and complete it *before* continuing with the example below

Work through the Cluster Analysis dialog set-up in the following manner:

1. Select **Affymetrix Summary** from the **Show Data of Type** drop-down list.
2. Select **MouseSwimExprSet.norm** from the **Data** drop-down list.

3. Go to the **Filtering Options** tab and select the **Genes with maximum fold change greater than** checkbox and set the associated field to **8**. We want to limit the genes to only those with extreme fold change.
4. Click the **Recalculate** button to the right in the **Gene Sort Order Options** group. You should get 51 genes resulting from the filtering which will be done prior to clustering.

The resulting dialog tabs are displayed in Figures 8.3 and 8.4.

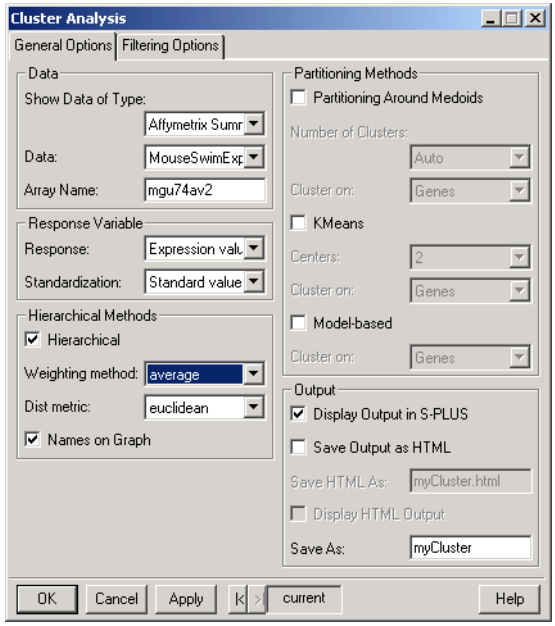


Figure 8.3: *General Options ready for simple hierarchical clustering of expression values.*

Standardization

The **Standardization** options in the **Response Variable** group allows three options for "standardizing" the data before clustering.

1. **None**: raw expression intensities are used.
2. **Median polish**:
3. **Standard values**: Values are standardized by subtracting off the mean and dividing by the standard deviation.

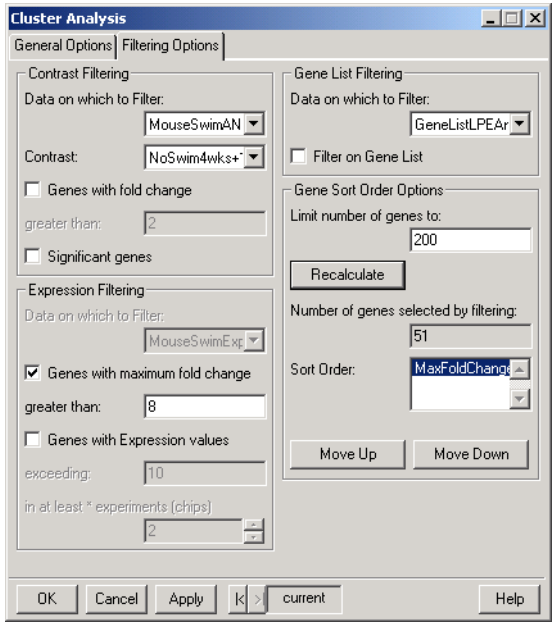


Figure 8.4: *Filtering Options ready for simple hierarchical clustering of expression values with maximum fold change greater than 8.*

The resulting output is displayed in Figure 8.5. The clustering dendrogram for expression values is displayed on the left side of the heat map; gene labels are on the right side. The dendrogram at the top of the page results from clustering on experimental conditions. The legend at the bottom show the relationship between colors in the heat map and (logged) expression intensity values.

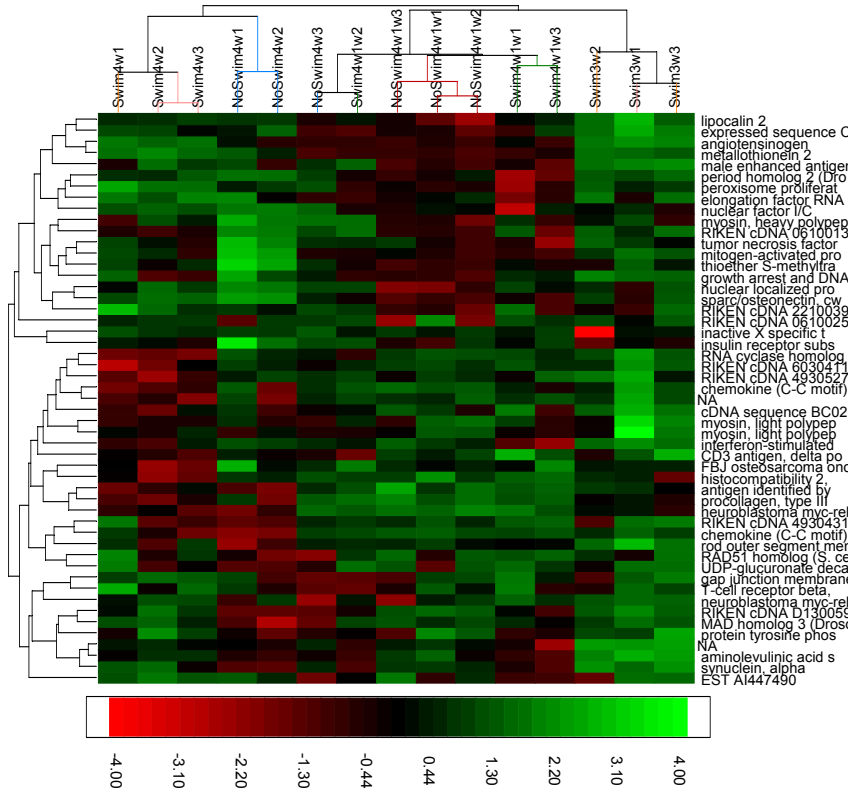


Figure 8.5: *Hierarchical clustering output for the swimming mouse example expression data. The clustering dendrogram is displayed on the left side of the graph with gene tables on the right. The dendrogram at the top corresponds to the experimental conditions. The legend across the bottom indicates the values of the expression intensities displayed in the heat map.*

Partitioning

We can now change the clustering method simply by selecting one of the partitioning methods and re-running the analysis. Partitioning around medoids automatically estimates the number of clusters and generates three graphics.

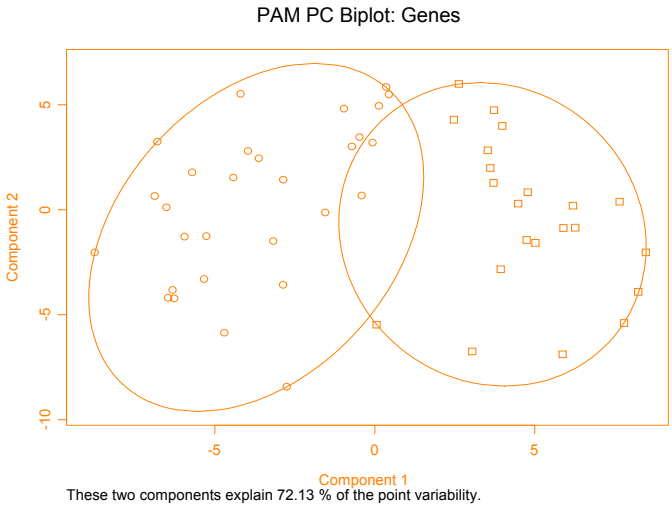


Figure 8.6: *Partitioning around medoids plot of the first two principal components of expression values with clustering boundaries specified.*

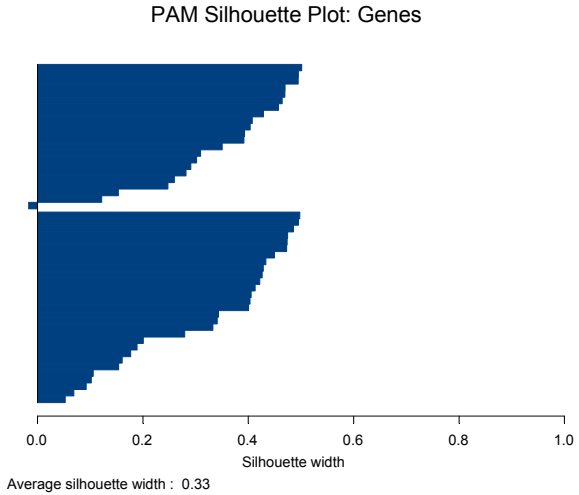


Figure 8.7: *Partitioning around medoids silhouette plot of expression values. Separation of the silhouettes for each group indicates good separation of clusters.*

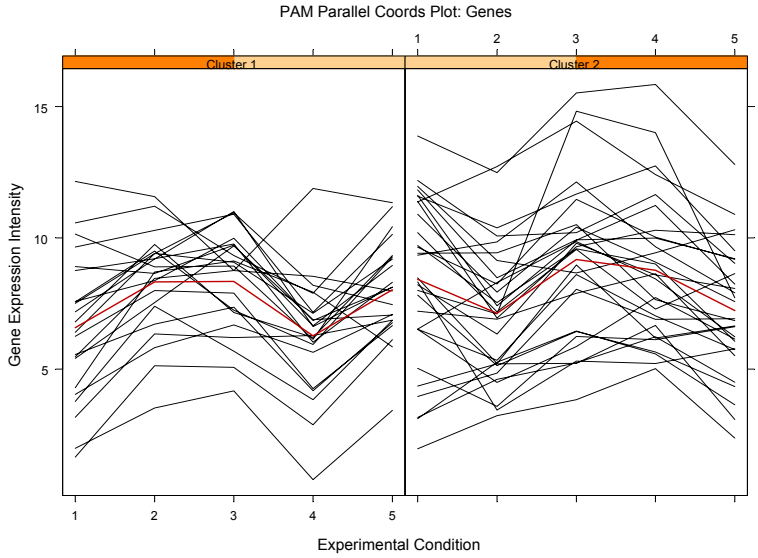


Figure 8.8: *Partitioning around medoids parallel coords plots expression values across experimental conditions for each cluster.*

Clustering Two-Channel Expression Data

Clustering two-channel data is not that much different from single-channel expression intensity data. The key difference is that you now have a choice of variables to cluster on. You can choose between

1. M, the $\log_2$ ratio of the two channels, or
2. both red and green channels.

In the second case, both channels are clustered simultaneously.

As an example, use one of the datasets created in Chapter 4, Examples: Two-Color Data. Proceed as follows:

1. Select data of type **Two Channel** from the **Show Data of Type** drop-down list and select one of the two-channel datasets from the **Data** drop-down list.
2. Select one of **M** or **red and green channels** from the **Response** drop-down list in the **Response Variable** group.

3. Go to the **Filtering Options** tab and select the **Genes with maximum fold change greater than** checkbox and set the associated field to **8**. We want to limit the genes to only those with extreme fold change.
4. Select a clustering method and click **Apply** or **OK** to run the analysis.

The resulting plots are similar to those for Affymetrix Summary data.

Clustering Significant Genes From An ANOVA

The other data type option from the **Show Data of Type** drop-down list is **ANOVA-DiffExprTest**. This option allows you to cluster on only those genes that are significant in an analysis. As an example, use the ANOVA object created during the analysis of the Swimming Mouse data in Chapter 2, Examples: Affymetrix MAS Data. We named the ANOVA object **MouseANOVANoSwim4wksBH**. Proceed as follows:

1. Select **ANOVA-DiffExprTest** from **Show Data of Type** drop-down list.
2. Select **MouseANOVANoSwim4wksBH** from the **Data** drop-down list.
3. On the **Filtering Options** tab, in the **Contrast Filtering** group, from the **Contrast** drop-down list, select **Swim3wks-NoSwim4wks**
4. Select the S**ignificant genes** checkbox in **Contrast Filtering** group.
5. Click the Recalculate button in the **Gene Sort Order Options** group. You should see 15 listed.
6. Now on the **General Options** tab select the clustering method of choice and run the analysis.

The example output below results from partitioning around medoids clustering.

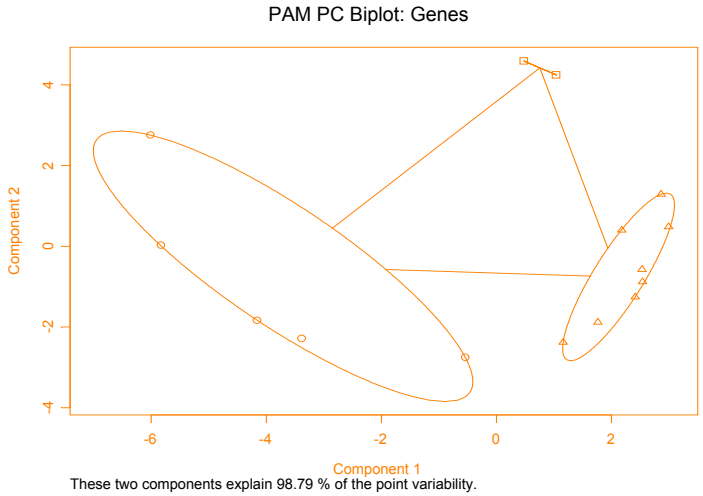


Figure 8.9: *Biplot resulting from partitioning around medoids for only the significant genes of one contrast in the* ***MouseANOVANoSwim4wksBH*** *object.*

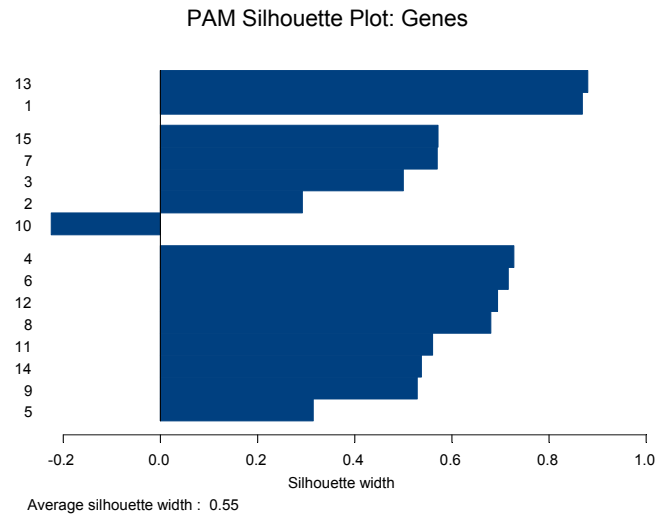


Figure 8.10: *Silhouette plot resulting from partitioning around medoids for only the significant genes of one contrast in the* ***MouseANOVANoSwim4wksBH*** *object.*

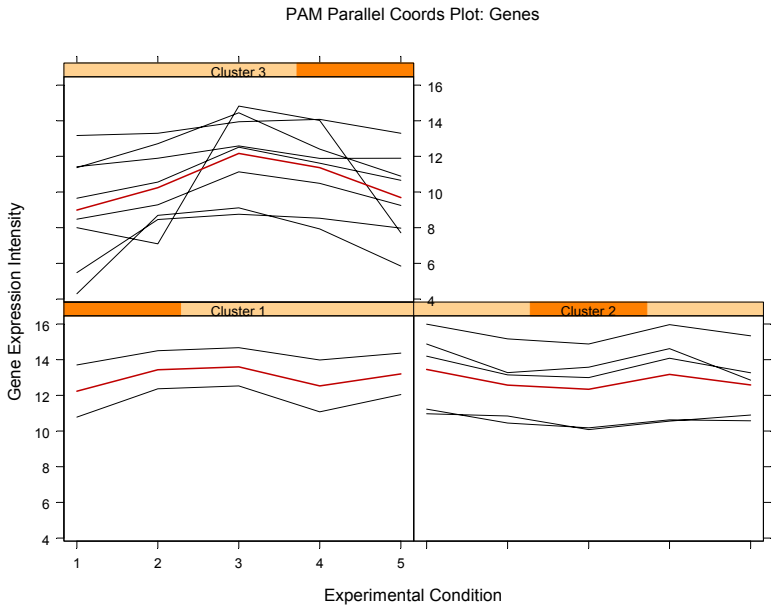


Figure 8.11: *Parallel Coords plot resulting from partitioning around medoids for only the significant genes of one contrast in the* ***MouseANOVANoSwim4wksBH*** *object.*

EXAMPLES FROM THE COMMAND LINE

The methods described in section Hierarchical Methods on page 304 and section Partitioning Methods on page 308 are available from the S-PLUS command line. The methods are provided in several flavors through different S-PLUS functions. To become fluent in clustering in S-PLUS takes a bit of practice but it's well worth the effort. The flexibility and richness of the S-PLUS language extends the current capabilities of the GUI.

Available Methods

A variety of partitioning and hierarchical cluster analysis methods are available in S-PLUS, including a library of algorithms described in Kaufman and Rousseeuw (1990).

Partitioning Methods

The partitioning methods include K-means, partitioning around medoids, and a fuzzy clustering method in which probability of membership of each class is estimated. These methods are available through the following S-PLUS functions:

`kmeans`: K-means clustering

`pam`: partitioning around medoids

`clara`: partitioning around medoids for large data sets

The key difference between `kmeans` and `pam` is that `kmeans` uses a mean as the center of each cluster and `pam` uses an actual data point as the center.

Hierarchical Methods

The hierarchical methods include agglomerative methods (which start from individual points and successively merge clusters until one cluster representing the entire dataset remains) and divisive methods (which consider the whole dataset and split it until each object is separate).

The available agglomerative methods are

`hclust`: Performs hierarchical clustering on a distance or similarity structure.

`agnes`: Similar to `hclust` but the algorithm yields a measure of the amount of clustering structure found.

`mclust`: Model-based clustering with many options.

The available divisive methods are

`diana`: General divisive clustering.

`mona`: Divisive clustering for only binary variables.

Distance Metrics

All cluster methods are very sensitive to the choice of distance or dissimilarity *between points* (i.e., samples or genes). S-PLUS includes two commonly used functions for creating distances or dissimilarities between points, namely `dist` and `daisy`. The correlation function, `cor`, may also be used, and `1 - cor(x)` produces a matrix representing the dissimilarities between columns (samples) of a matrix $\boldsymbol{x} = (x_{ij})$.

The `dist` function simply constructs distances between rows as Euclidean, Manhattan, maximum, and binary. If the data are normalized with mean equal to zero and variance equal to one prior to calling `dist`, the resulting matrix is equivalent to a dissimilarity matrix produced using `cor`.

Example: Lymphoma Classification

In cancer diagnostics, there is considerable interest in subpopulations of cancer tissue samples. For example, distinct subpopulations identified within the collection of samples may have different etiologies and may be candidates for different clinical interventions.

Alizadeh et al. (2000) characterized variability in gene expression among tumors in lymphoma patients using a customized cDNA lympho-chip. This chip included genes expressed in lymph cells and genes that play an important role in cancer. They ran samples from the three most common adult lymphomas on the lympho-chip: namely diffuse large B-cell lymphoma (DLBCL), follicular lymphoma (FL), and chronic lymphocytic leukemia (CLL), and a variety of other lymphoma and leukemia cell lines. Each chip had a reference sample, with cy5 labeling used for the experimental samples and cy3 for the reference samples. Alizadeh et al. (2000) identified two distinct subtypes of DLCBL from a hierarchical cluster analysis of the resulting data; the relevant heat map and dendrogram from this analysis are given in Figure 3a of Alizadeh et al. (2000).

Note that Alizadeh et al. (2000) focused their attention on B cell differentiation genes based on visual examination of hierarchical cluster analysis and heat map visualization of 96 samples run on arrays of more than 10,000 genes. We do not recommend this qualitative approach; rather, we suggest genes be included in cluster analyses based on their differential expression according to a reliable statistical hypothesis-testing procedure.

We provide two analyses of the subset of data presented in Figure 3a of Alizadeh et al. (2000) using hierarchical and partitioning cluster routines. We actually use the data as summarized by Cluster (Eisen et al., 1998) and prepared for viewing in TreeView (Eisen et al., 1998). Note that this is not the actual raw data. We treat it as raw data to show the cluster methods in S-PLUS, but the resulting output should not be directly compared with fig3a of Alizadeh et al. (2000).

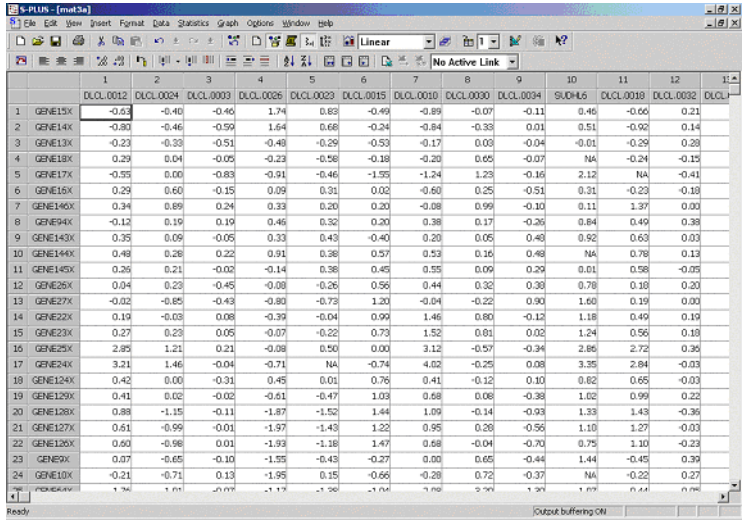
Hierarchical Clustering Example

Our hierarchical cluster method (`hclust`) uses a group average between-cluster dissimilarity measure. The partitioning method uses the partitioning around medoids method (`pam`).

We begin by importing the data described above. The data can be downloaded from

http://llmpp.nih.gov/lymphoma/data.shtml

From the **Figure 3** link, download the file named **figure3a.cdt** into the **splus62/module/ArrayAnalyzer/examples** directory. The first two lines in the following code will import the data and create a data frame, which we call `mat3a`.



		1	2	3	4	5	6	7	8	9	10	11	12	13
		DLCL.0012	DLCL.0024	DLCL.0003	DLCL.0026	DLCL.0023	DLCL.0015	DLCL.0010	DLCL.0030	DLCL.0034	SUDHL6	DLCL.0018	DLCL.0032	DLCL
1	GENE15X	-0.53	-0.40	-0.46	1.74	0.83	-0.49	-0.89	-0.07	-0.11	0.46	-0.66	0.21	
2	GENE14X	-0.80	-0.46	-0.59	1.64	0.68	-0.24	-0.84	-0.33	0.01	0.51	-0.92	0.14	
3	GENE13X	-0.23	-0.33	-0.51	-0.48	-0.29	-0.53	-0.17	0.03	-0.04	-0.01	-0.29	0.28	
4	GENE18X	0.29	0.04	-0.05	-0.23	-0.58	-0.18	-0.20	0.65	-0.07	NA	-0.24	-0.15	
5	GENE17X	-0.55	0.00	-0.83	-0.91	-0.46	-1.55	-1.24	1.23	-0.16	2.12	NA	-0.41	
6	GENE16X	0.29	0.60	-0.15	0.09	0.31	0.02	-0.60	0.25	-0.51	0.31	-0.23	-0.18	
7	GENE146X	0.34	0.89	0.24	0.33	0.20	0.20	-0.08	0.99	-0.10	0.11	1.37	0.00	
8	GENE94X	-0.12	0.19	0.19	0.46	0.32	0.20	0.38	0.17	-0.26	0.84	0.49	0.38	
9	GENE143X	0.35	0.09	-0.05	0.33	0.43	-0.40	0.20	0.05	0.48	0.92	0.63	0.03	
10	GENE144X	0.48	0.26	0.22	0.91	0.38	0.57	0.53	0.16	0.48	NA	0.78	0.13	
11	GENE145X	0.26	0.21	-0.02	-0.14	0.38	0.45	0.55	0.09	0.29	0.01	0.58	-0.05	
12	GENE26X	0.04	0.23	-0.45	-0.08	-0.26	0.56	0.44	0.32	0.39	0.78	0.18	0.20	
13	GENE27X	-0.02	-0.85	-0.43	-0.80	-0.73	1.20	-0.04	-0.22	0.90	1.60	0.19	0.00	
14	GENE22X	0.19	-0.03	0.08	-0.39	-0.04	0.99	1.46	0.80	-0.12	1.18	0.49	0.19	
15	GENE23X	0.27	0.23	0.05	-0.07	-0.22	0.73	1.52	0.81	0.02	1.24	0.56	0.18	
16	GENE25X	2.85	1.21	0.21	-0.08	0.50	0.00	3.12	-0.57	-0.34	2.86	2.72	0.36	
17	GENE24X	3.21	1.46	-0.04	-0.71	NA	-0.74	4.02	-0.25	0.08	3.35	2.84	-0.03	
18	GENE124X	0.42	0.00	-0.31	0.45	0.01	0.76	0.41	-0.12	0.10	0.82	0.65	-0.03	
19	GENE129X	0.41	0.02	-0.02	-0.61	-0.47	1.03	0.68	0.08	-0.38	1.02	0.99	0.22	
20	GENE128X	0.88	-1.15	-0.11	-1.87	-1.52	1.44	1.09	-0.14	-0.93	1.33	1.43	-0.36	
21	GENE127X	0.61	-0.99	-0.01	-1.97	-1.43	1.22	0.95	0.28	-0.56	1.10	1.27	-0.03	
22	GENE126X	0.60	-0.98	0.01	-1.93	-1.18	1.47	0.68	-0.04	-0.70	0.75	1.10	-0.23	
23	GENE9X	0.07	-0.65	-0.10	-1.55	-0.43	-0.27	0.00	0.65	-0.44	1.44	-0.45	0.39	
24	GENE10X	-0.21	-0.71	0.13	-1.95	0.15	-0.66	-0.28	0.72	-0.37	NA	-0.22	0.27	

Figure 8.12: *Imported data from Figure 3a of Alizadeh et al. (2000). Note that this is not the actual raw data, but rather data as summarized by Cluster (Eisen et al., 1998) and prepared for viewing in TreeView (Eisen et al., 1998). We treat it as raw data to show the cluster methods in S-PLUS, but the resulting output should not be directly compared with fig3a of Alizadeh et al. (2000).*

We then standardize this data frame, calculate the distances between points in the column and row spaces, and fit the hierarchical cluster models. Note that since the data are normalized with mean zero and variance one prior to calling `dist`, the resulting matrix is equivalent to a dissimilarity matrix produced using `cor`.

```
> module(ArrayAnalyzer)
> fileName <- file.path(getenv("SHOME"),
      "module","ArrayAnalyzer","examples","figure3a.cdt")
> mat3a <- importData(fileName, rowNamesCol=1,
     colNameRow=1, drop=c(2:4), startRow=3, type="ASCII")
> stand.norm <- function(x)
    {
       (x- mean(x, na.rm=T))/
           sqrt(var(x,na.method="available"))
    }
> aliz.cmat <- apply(mat3a,1,stand.norm)

# cluster rows
```

```
> aliz.dist1 <- dist(t((aliz.cmat)))
> aliz.hclust1 <- hclust(dist=aliz.dist1,method="average")

# cluster cols
> aliz.dist2 <- dist(as.matrix(aliz.cmat))
> aliz.hclust2 <- hclust(dist=aliz.dist2,method="average")

# color 6 = GC B like; color 5 = Activated B like;
# color 1 = GC centroblasts
> array3a.colors <-
     c(rep(6,16),rep(1,2),rep(6,6),rep(5,23))

# plot heat map and dendrograms
> par(mai=c(0,0,0,0),omi=c(0,2.7,1.4,1.1))
   image(aliz.cmat[aliz.hclust2$order,aliz.hclust1$order],
   axes=F,bty="n")
> par(new=T,omi=c(6.55,2.75,0,1.15))
> plclust2.fn(aliz.hclust2,cex=1,rotate.me=F,lty=1,
     colors=array3a.colors[aliz.hclust2$order])
> par(new=T,omi=c(0.02,0.95,1.42,7.75))
> plclust2.fn(aliz.hclust1,cex=1,rotate.me=T,lty=1)
```

The cluster analysis can also be done from the S-PLUS menu system using **Statistics ▶ Cluster Analysis ▶ Agglomerative Hierarchical**, although this does not produce the heat map with overlaid dendrograms as visual output like the above code snippet.

Note that the above code snippet can be easily saved as an S-PLUS function for repeated use as follows:

```
> cluster.heat <- function(cluster.data,
  sample.colors = rep(1,dim(cluster.data[,1])))
{
     stand.norm <- function(x)
     { (x - mean(x, na.rm=T))/
        sqrt(var(x, na.method="available"))
     }
  cmat <- apply(cluster.data,1,stand.norm)
  # cluster rows
  dist1 <- dist(t(as.matrix(cmat)))
  hclust1 <- hclust(dist=dist1,method="average")
  # cluster cols
  dist2 <- dist(as.matrix(cmat))
```

```
  hclust2 <- hclust(dist=dist2,method="average")
  # plot heat map and dendrograms
  par(mai=c(0,0,0,0),omi=c(0,2.7,1.4,1.1))
  image(cmat[hclust2$order,hclust1$order], axes=F,bty="n")
  par(new=T,omi=c(6.55,2.75,0,1.15))
  plclust2.fn(hclust2,cex=1,rotate.me=F,lty=1,
      colors=sample.colors[hclust2$order])
  par(new=T,omi=c(0.02,0.95,1.42,7.75))
  plclust2.fn(aliz.hclust1,cex=0.1,rotate.me=T,lty=1)
}
```

This function could then be called to analyze the Alizadeh data as follows. Of course, the function could do with some error checking if it were planned to be used by others!

```
> cluster.heat(mat3a, sample.colors = c(rep(6,16),
   rep(1,2), rep(6,6), rep(5,23)))
```

Note that this function and the function `plclust2.fn`, which allows a dendrogram to be rotated and easily laid next to a heat map, are included in the S+ARRAYANALYZER module.

Results of the hierarchical cluster analysis are presented below in Figure 8.13. Overall, we produce a clustering result similar to that of Alizadeh et al. (2000). Note that the data we worked with are not the actual raw data. We treat it as raw data to show the cluster methods in S-PLUS, but the resulting output should not be directly compared with figure 3a of Alizadeh et al. (2000).

We have ordered the columns based on the default rule in S-PLUS, namely that, at each merge, the subtree with the tightest cluster is placed to the left. This is the opposite of the ordering used by the package Cluster (Eisen et al., 1998), which was used by Alizadeh et al. (2000). The columns of our heat map are thus in approximately reverse order to those presented in Alizadeh et al. (2000). Also note that individuals within nodes are paired by their original order in S-PLUS, while this ordering is at random in the package Cluster (Eisen et al., 1998). Further, Alizadeh et al. (2000) use a weighting function that is not well documented in the Cluster (Eisen et al., 1998) manual.

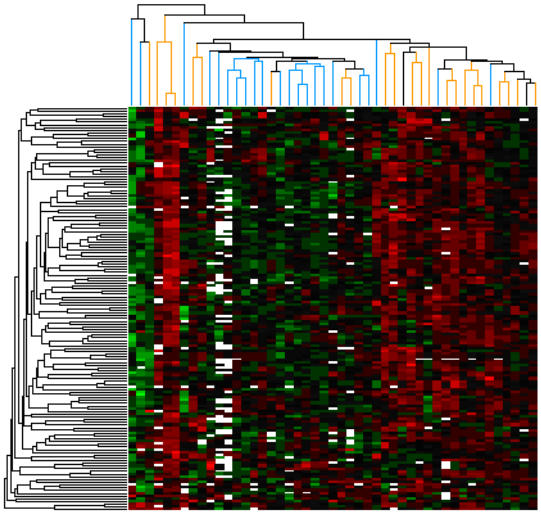


Figure 8.13: *Heat map and dendrogram based on data from Alizadeh et al. (2000). Note that this is not the actual raw data, but rather data as summarized by Cluster (Eisen et al., 1998) and prepared for viewing in TreeView (Eisen et al., 1998). We treat it as raw data to show the cluster methods in S-PLUS, but the resulting output should not be directly compared with fig3a of Alizadeh et al. (2000).*

Partitioning Around Medoids Example

Since Alizadeh et al. (2000) were interested in identifying two specific subpopulations within the DLBCL samples, they may have used a partitioning clustering method. We use the partitioning around medoids method (`pam`). This analysis provides some evidence for the existence of two subpopulations rather than three, four or five subpopulations based on average silhouette width. The average silhouette width for two subpopulations is 0.19 compared to 0.15 and 0.08 for three and four subpopulations. However, absolute values of the average silhouette width are fairly small in all cases.

The partitioning around medoids analysis and graphical summaries are presented in Figure 8.14 and Figure 8.15. Figure 8.14 shows the two clusters projected onto biplot of the first two principal components. A silhouette plot for two subpopulations is provided in Figure 8.15:

```
# partitioning - 2 classes (compare to 3 and up)
mat3a.2.pam <- pam(t(mat3a),2)
plot(mat3a.2.pam)
```

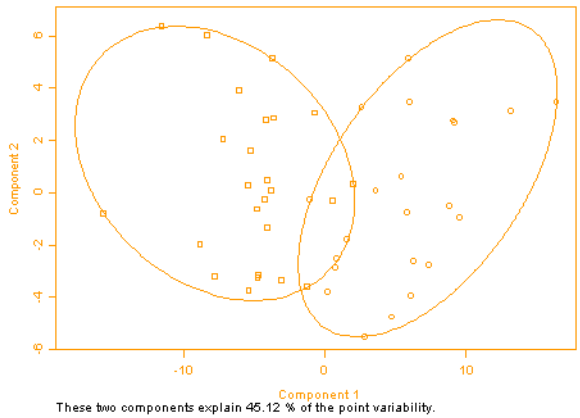


Figure 8.14: *Partitioning around medoids analysis summary for the Alizadeh et al. (2000) lymphoma data. The two clusters are projected onto a biplot of the first two principal components.*

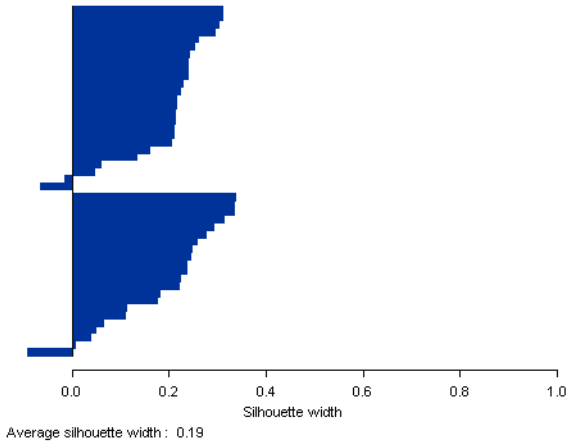


Figure 8.15: *Silhouette plot for the two subpopulation partitioning around medoids clustering for the Alizadeh et al. (2000) lymphoma data.*

REFERENCES

Alizadeh A.A., Eisen M.B., Davis R.E., Ma C., Lossos I.S., Rosenwald A., Boldrick J.C., Sabet H., Tran T., Yu X., Powell J.I., Yang L., Marti G.E., Moore T., Hudson T. Jr., Lu L., Lewis D.B., Tibshirani R., Sherlock G., Chan W.C., Greiner T.C., Weisenburger D.D., Armitage J.O., Warnke R., Levy R., Wilson W., Grever M.R., Byrd J.C., Botstein D., Brown P.O., Staudt L.M. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature,* 403:503-511.

Eisen M.B., Spellman P.T., Brown P.O., Botstein D. (1998). Cluster analysis and display of genome-wide expression patterns. Proceedings of National Academic Sciences USA, 95(25):14863-14868.

Kaufman, L., Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons: New York.

Kerr M.K., Churchill G.A. (2001). Bootstrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. Proceedings of National Academic Sciences USA, 98:8961-8965.

Ross D.T., Scherf U., Eisen M.B., Perou C.M., Rees C., Spellman P., Iyer V., Jeffrey S.S., Van de Rijn M., Waltham M., Pergamenschikov A., Lee J.C., Lashkari D., Shalon D., Myers T.G., Weinstein J.N., Botstein D., Brown PO (2000). Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics* 24(3): 227-35.

Scherf U., Ross D.T., Waltham M., Smith L.H., Lee J.K., Kohn K.W., Reinhold W.C., Myers T.G., Andrews D.T., Scudiero D.A., Eisen M.B., Sausville E.A., Pommier Y., Botstein D., Brown P.O., Weinstein J.N. (2000). A cDNA microarray gene expression database for the molecular pharmacology of cancer. *Nature Genetics* 24(3):236-244.

Yeung, K.Y., Fraley, C., Murua, A., Raftery, A., Ruzzo, W.L. (2001). *Model-Based Clustering and Data Transformations for Gene Expression Data*. Technical Report #396, Department of Statistics, University of Washington: Seattle, WA.

ANNOTATION AND GENE LIST MANAGEMENT

9

Annotation and Gene List Management Functionality **332**

Annotation Libraries **334**

Annotation from Graphical and Tabular Reports: GenBank Sites 338

Annotation from the GUI: GenBank and Other Browser Metadata Lookups 341

Annotation from the GUI: Genelist Analysis 349

Annotation Using Affymetrix IDs 350

Annotation Using LocusLink IDs 356

Annotation Using OntoExpress 357

Annotation from Command Line Scripting: GenBank Metadata Lookups 359

Filtering Genes Based on GO Categories 363

ANNOTATION AND GENE LIST MANAGEMENT FUNCTIONALITY

S+ARRAYANALYZER primarily uses the annotation metadata maintained by the Bioconductor project, repackaged as S-PLUS libraries and data objects for fast lookup and display. Bioconductor maintains Affymetrix chip-specific and general annotation data packages; these data packages and the process for creating them are described in detail below.

The Affymetrix chip-specific and general annotation data libraries are used by S+ARRAYANALYZER to do the following:

- Annotate graphical and tabular reports from statistical analyses using gene lookup metadata sites, such as LocusLink and Entrez.
- Annotate gene lists derived from the statistical analyses via metadata repositories such as LocusLink, Entrez, Pubmed, AmiGO and Source.
- Connect to gene list analysis sites such as Onto-Express and DAVID/EASE, and initiate gene list analyses (e.g., gene function enrichment and identification of GO categories that are overrepresented in gene lists derived from statistical analyses).
- Subset microarray datasets according to GO categories prior to (differential expression) analysis.

The annotation libraries contain S-PLUS objects (lists) for each metadata type, typically with names corresponding to probe-ids for the probes on the array and entries corresponding to the metadata mappings. The probe-ids on an array are typically unique (for any manufacturer). Note that when the identifiers from one source are linked to the identifiers from another there does not need to be a one-to-one relationship. For example, several different Affymetrix probe-ids correspond to a single LocusLink identifier. Thus, when mapping Affymetrix to LocusLink, we have no problem, but mapping LocusLink to Affymetrix requires a mechanism for dealing with the multiplicity of matches.

There is a great deal of annotation data available for any given gene. Examples include LocusLink, Unigene, chromosome number, chromosomal location (cytoband or bp), KEGG pathway information, and the Gene Ontology (GO) categorizations. The Annotation libraries for arrays in common use are included in the S+ARRAYANALYZER installation media. All of the Annotation libraries are available from the data libraries link in the S+ARRAYANALYZER Web site:

http://www.insightful.com/support/ArrayAnalyzer/DataLibs/README.html

ANNOTATION LIBRARIES

The Affymetrix chip-specific annotation libraries in S+ARRAYANALYZER are as follows:

- **hgu133a, hgu133b**, hgu133plus2
- **hgu95av2**, hgu95b, hgu95c, hgu95d, hgu95e, hu6800
- **moe430a, moe430b**, moe430v2
- mgu74a, **mgu74av2**, mgu74b, mgu74bv2, mgu74c, mgu74cv2
- **rae230a**, **rae230b**
- **rgu34a**, rgu34b, rgu34c

The general annotation libraries in S+ARRAYANALYZER are as follows:

- **GO**
- **KEGG**
- humanCHRLOC
- humanLLMappings
- mouseCHRLOC
- mouseLLMappings
- ratCHRLOC
- ratLLMappings

The libraries in bold above are included in the S+ARRAYANALYZER installation CD. All of the Annotation libraries are available from the S+ARRAYANALYZER data libraries Web site (referenced on the previous page).

The Annotation libraries are automatically attached when you attach S+ARRAYANALYZER. To manually attach the Affymetrix chip-specific libraries and use the libraries in scripting at the command line, enter

```
> library(<chipname>AnnoData)
```

Attaching an annotation library makes all the S-PLUS annotation objects for that library available in the current S-PLUS session. For example, the S-PLUS annotation objects available in the hgu95av2 library are shown in Table 9.1.

Table 9.1: *S-PLUS Affymetrix chip-specific library objects.*

S-PLUS Annotation object	**Description**
`hgu95av2ACCNUM`	Maps probe ids to GenBank accession number or user specified ids
`hgu95av2CHRLENGTHS`	A vector containing the lengths (in base pairs) of chromosomes
`hgu95av2CHRLOC`	Maps probe ids to chromosomal locations
`hgu95av2CHR`	Maps probe ids to chromosome numbers
`hgu95av2ENZYME2PROBE`	Maps enzyme EC numbers to probe ids
`hgu95av2ENZYME`	Maps probe ids to enzyme EC numbers
`hgu95av2GENENAME`	Maps probe ids to gene names
`hgu95av2GO2ALLPROBES`	An annotation data file for GO2ALLPROBES
`hgu95av2GO2PROBE`	Maps GO ids to probe ids with specific association
`hgu95av2GO`	Maps probe ids to GO data (ids, evidence code, and ontology)
`hgu95av2GRIF`	Maps probe ids to the unique PubMed id

Table 9.1: *S-PLUS Affymetrix chip-specific library objects. (Continued)*

S-PLUS Annotation object	Description
hgu95av2HGID	Maps probe ids to internal HomoloGenIds
hgu95av2LOCUSID	Maps probe ids to LocusLink ids
hgu95av2MAP	Maps probes to cytobands
hgu95av2NM	Maps probe ids to RefSeq accession numbers for mRNA records
hgu95av2NP	Maps probe ids to RefSeq accession numbers for protein records
hgu95av2OMIM	Maps probe ids to MIM numbers
hgu95av2ORGANISM	The name of the organism (hgu95av2 in this case)
hgu95av2PATH2PROBE	Maps KEGG pathway ids to probe ids
hgu95av2PATH	Maps probe ids to KEGG pathway ids
hgu95av2PMID2PROBE	Maps PubMed ids to probe ids

These annotation objects contain some redundancies; the objects most commonly used include <>ACCNUM, <>LOCUSID, <>GENENAME, <>GO, <>PATH, <>PMID, and <>UNIGENE.

The general annotation libraries (e.g., GOAnnoData, KEGGAnnoData) may be manually attached as follows:

```
> library(<chipname>AnnoData)
```

S-PLUS objects for some of these libraries are shown in Table 9.2.

Table 9.2: *S-PLUS general annotation library objects.*

S-PLUS Annotation object	**Description**
GOBPCHILDREN GOBPPARENTS GOCCCHILDREN GOCCPARENTS GOGO2LL GOMFCHILDREN GOMFPARENTS GOTERM	Annotation data files for GO
KEGGENZYMEID2GO KEGGEXTID2PATHID KEGGGO2ENZYMEID KEGGPATHID2EXTID KEGGPATHID2NAME KEGGPATHNAME2ID	Annotation data files for KEGG
<species>CHRLOC<x>START <species>CHRLOC<x>END <species>CHRLOCCYTOLOC <species>CHRLOCLOCUSID2CHR	Chromosome location start and end for <x> = 1,...,22, x,y for <species> = human, rat and mouse Supporting cytoband and locus link information
<species>LLMappingsGO2LL <species>LLMappingsLL2GO <species>LLMappingsLL2PMID <species>LLMappingsLL2UG <species>LLMappingsPMID2LL <species>LLMappingsUG2LL	Locus Link mappings for <species> = human, rat and mouse
homology*	Various homology annotation data files and mappings

For non-Affymetrix two-channel chips, one can construct similar S-PLUS annotation objects for use in S+ARRAYANALYZER by using the **<layoutFileName>** rather than the Affymetrix chip name as the

prefix in the annotation object. For example, if the layout file name used in the data import is **AgilHuv2**, the corresponding accession numbers could be simply inserted into an S-PLUS data object `AgilHuv2ACCNUM`. The format for these objects is an S-PLUS list with names corresponding to (unique) probe ids and entries the actual accession numbers. For reverse mapping lookups, the values of the annotation element are the names (keys) and the entries are the probe ids.

Using this approach, S-PLUS annotation objects can be simply built for non-Affymetrix and custom arrays. Once these S-PLUS annotation objects are in place, they are automatically used by the graphical and tabular reports in S+ARRAYANALYZER.

Annotation from Graphical and Tabular Reports: GenBank Sites

In the first example, we show how the S+ARRAYANALYZER default tabular and graphical plots are linked to LocusLink and UniGene Web sites for simple metadata annotation. This example uses the Melanoma data (Fox et. al., 2001), and revisits results from reading in the **cel** files, summarizing with `RMA`, and performing differential expression testing using `LPEtest` with Bonferroni FWER control. Refer to Chapter 3, Examples: Affymetrix Probe-Level Data for details on this process.

Three of the tabular/graphical reports from the LPE analysis include annotation metadata links:

1. Top 15 Genes
2. Volcano plot
3. Heat map

We first review these reports, and then show how to link to additional annotation information from the **Annotation** dialog and via the command line scripting environment.

The *volcano* plot shows the genes' p-value v. fold-change. Each gene is active so you can click on a gene to access annotation information using its LocusLink ID or accession number. Other mappings can be added to this menu using the `menu.additem()` function in S-PLUS.

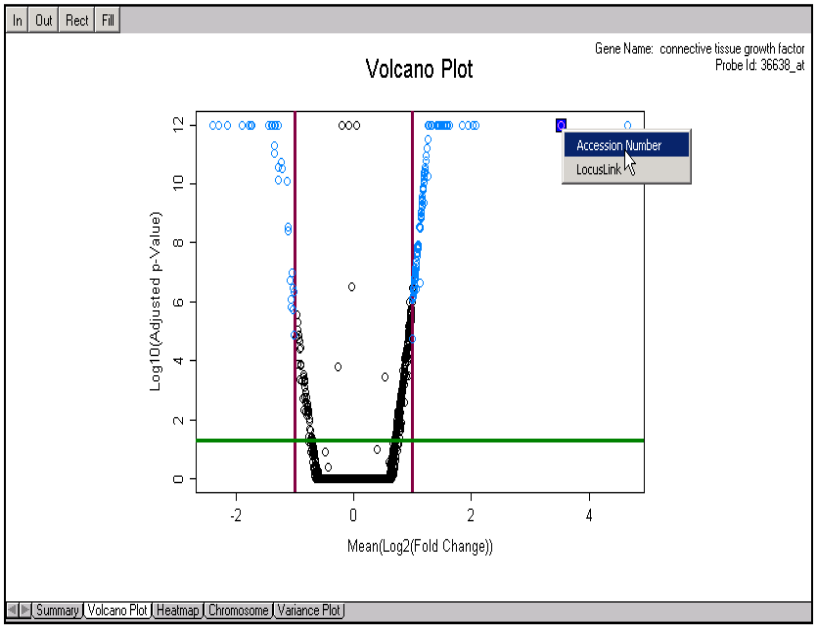


Figure 9.1: *A volcano plot, which is the logarithm of p-value versus fold change. Points above the horizontal line are hyperlinked to annotation databases.*

The *heat map* plot, shown in Figure 9.2, shows a two-way layout of the most differentially expressed genes along the vertical axis versus the experimental conditions on the horizontal axis. This graph is also hyperlinked to access annotation information using the genes' LocusLink ID or accession number.

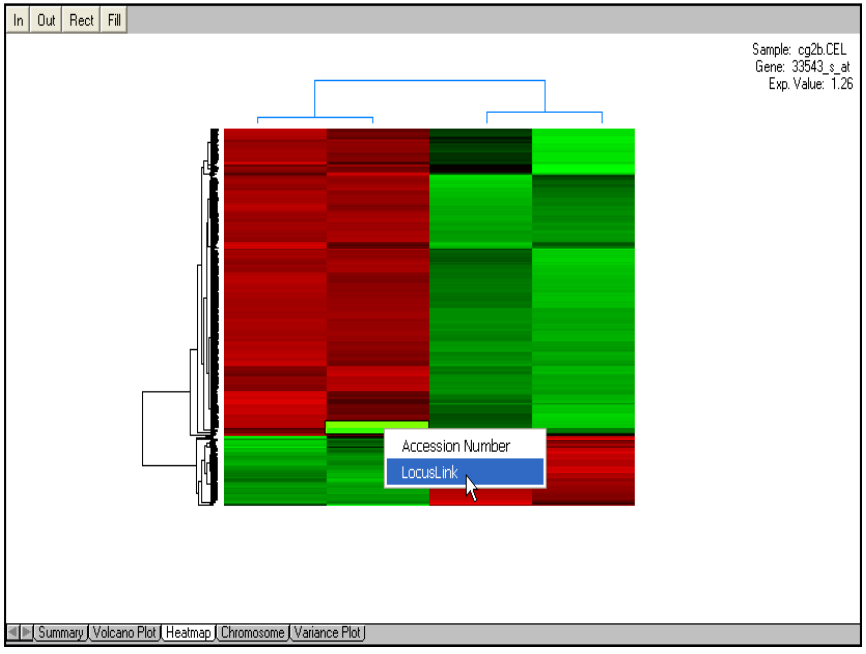


Figure 9.2: *A heat map plot shows differentially expressed genes as a function of experimental conditions. The map is hyperlinked to annotation databases.*

Clicking one of the hyperlinked points in the **Top 15 Genes Summary** plot, the volcano plot, or the heat map pops up a menu for selecting the database to query for annotation information. Selecting an entry in this menu opens an HTML page in your default Web browser displaying a brief description of the gene with hyperlinks to more detailed information. Figure 9.3 shows an example page from LocusLink with annotation for one of the differentially expressed genes from the volcano plot.

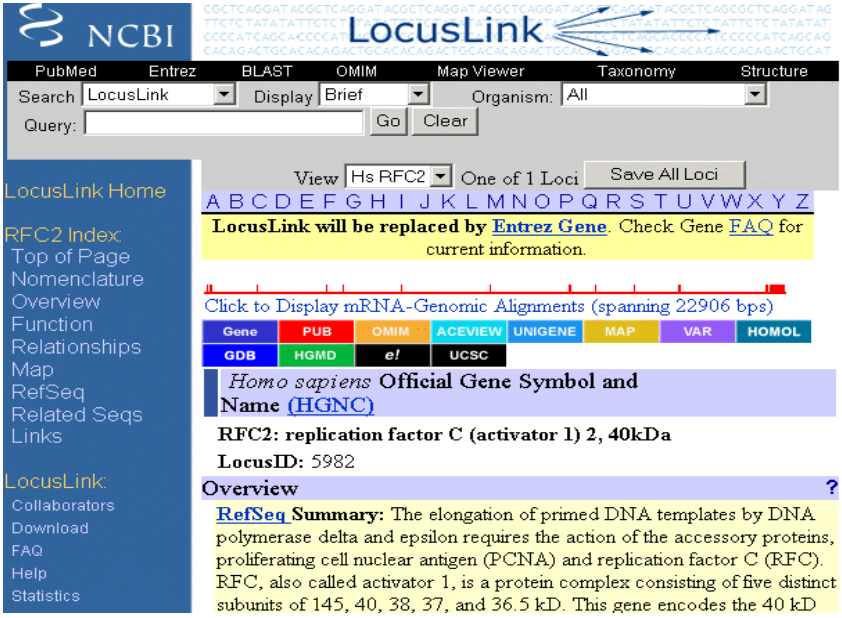

NCBI
LocusLink
PubMed Entrez BLAST OMIM Map Viewer Taxonomy Structure
Search LocusLink Display Brief Organism: All
Query: Go Clear
LocusLink Home
RFC2 Index:
Top of Page
Nomenclature
Overview
Function
Relationships
Map
RefSeq
Related Seqs
Links
LocusLink:
Collaborators
Download
FAQ
Help
Statistics
View Hs RFC2 One of 1 Loci Save All Loci
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
LocusLink will be replaced by Entrez Gene. Check Gene FAQ for current information.
Click to Display mRNA-Genomic Alignments (spanning 22906 bps)
Gene PUB OMIM ACEVIEW UNIGENE MAP VAR HOMOL
GDB HGMD e! UCSC
Homo sapiens Official Gene Symbol and Name (HGNC)
RFC2: replication factor C (activator 1) 2, 40kDa
LocusID: 5982
Overview
RefSeq Summary: The elongation of primed DNA templates by DNA polymerase delta and epsilon requires the action of the accessory proteins, proliferating cell nuclear antigen (PCNA) and replication factor C (RFC). RFC, also called activator 1, is a protein complex consisting of five distinct subunits of 145, 40, 38, 37, and 36.5 kD. This gene encodes the 40 kD

Figure 9.3: *Annotation information from LocusLink.*

Annotation from the GUI: GenBank and Other Browser Metadata Lookups

In this example, we show how to launch additional metadata Web sites (e.g., PubMed and GO, using gene lists derived from statistical analyses in S+ARRAYANALYZER). This example also uses the Melanoma data (Fox et. al., 2001) and picks up after the data have been read in through the GUI and analyzed for differential expression. Please refer to Chapter 3, Examples: Affymetrix Probe-Level Data for details on this process.

There are three parts to this analysis:

1. Choose the summary data analysis results for the melanoma experiment and the desired annotation sources.
2. Filter the genes to identify interesting genes for annotation.
3. Annotate these genes, which uses functions from the Annotate library.

For part 1, we use data that have been read in at the GUI and analyzed for differential expression. The object `cgLPEBon` is created by reading in the cel files, summarizing with RMA, and performing

differential expression testing using `LPEtest` with Bonferroni FWER control. Please refer to Chapter 3, Examples: Affymetrix Probe-Level Data to see how this object is created.

Figure 9.4 shows the **General Options** page of the **Annotation** dialog in S+ARRAYANALYZER with the differential expression test result object `cgLPEBon` selected, and the metadata sources LocusLink, Unigene, PubMed and GO Web site selected. Note that data types can be annotated through the **Annotation** dialog via Affymetrix summary data (MAS5/summarized CEL), two-channel data, differential expression test results (`DiffExprTest`), and the Gene Lists.

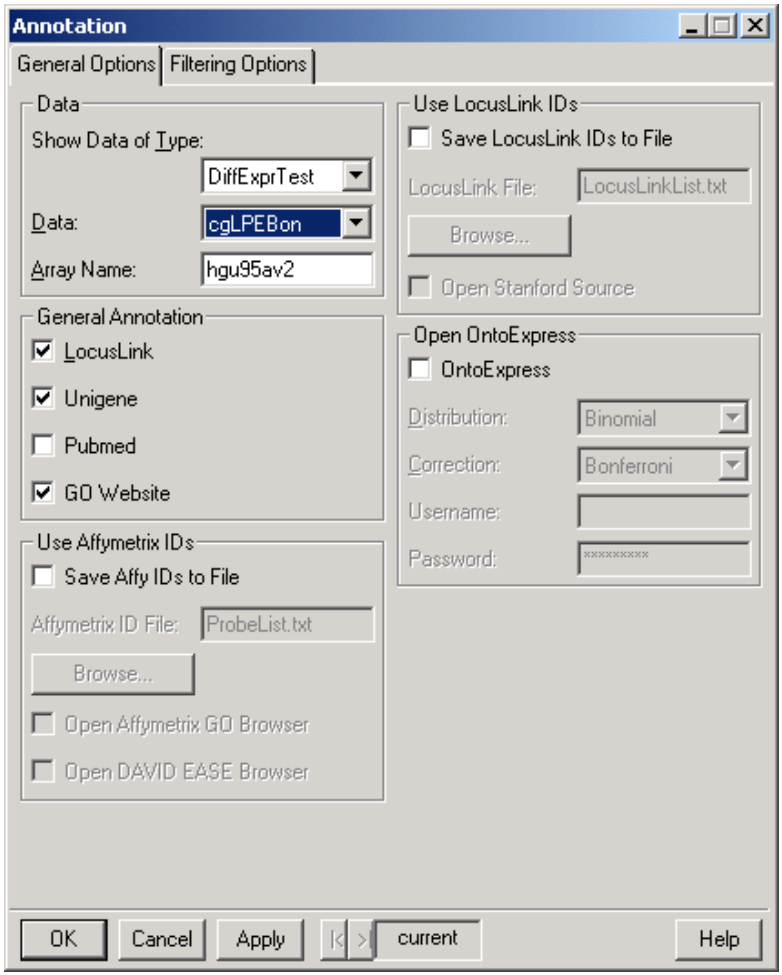


Figure 9.4: *The **General Options** page of the **Annotation** dialog. Options are highlighted from the **General Annotation** group.*

For part 2, we filter based on the p-value of the LPE differential expression test and the fold-change. Figure 9.5 below shows the **Filtering Options** page of the **Annotation** dialog, with the **Contrast Filtering** group activated and options fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ in this case).

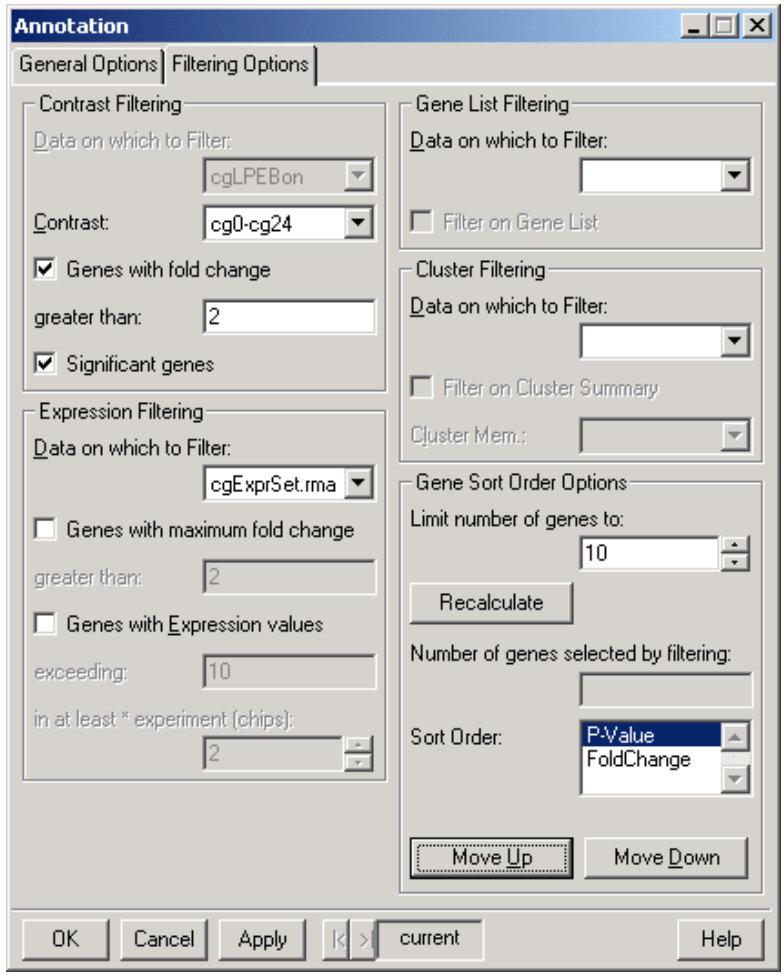


Figure 9.5: *The **Filtering Options** page of the **Annotation** dialog, with the **Contrast Filtering** group activated and options fold-change set to greater than 2, p-value set to significant (adjusted $p < 0.05$ in this case)*

The **Contrast Filtering** group is the natural filtering choice in this case since we are working with the results of a differential expression test. There are many filtering options available in the **Annotation** dialog:

Contrast Filtering

- Select genes with at least one fold-change greater than a user-specified number (e.g., 2).
- Significant genes or selects only genes declared significant for the test and FWER/FDR procedure/rate chosen in the differential expression test.
- The combination of these two options can be chosen to match the outside regions of the volcano plot, or a similarly filtered list of genes.

Expression Filtering

- Select genes with maximum fold-change greater than a user-specified number (e.g., 2)
- Select genes with an expression value exceeding a user-specified number A (e.g., 5, log2 scale) in at least k (user-specified) experiments. This is useful for filtering out genes that don't exhibit much expression in any experimental run.

Gene List Filtering

- Most of the analysis functions in S+ARRAYANALYZER can produce a gene list. Gene lists are managed in the **Gene List Management** dialog, where they can be intersectioned or unioned. These gene lists can be used to filter any of the data objects being annotated.

Cluster Filtering

- The partitioning cluster analyses (performed on genes) return a class membership for each gene. These class memberships can be used to filter any of the data objects being annotated.

Gene Sort Order Options

- Limit number of genes, which puts a cap on the gene list being annotated. This is particularly important for the GenBank databases; if you send too many queries in too short a time to GenBank, your IP address may be recorded, and you may be blocked from additional future use.
- Recalculate button, which allows you to monitor the number of genes in the gene list you are constructing through the filtering choices.
- Sort order, which allows you to sort the gene list you are constructing in order of the individual filters. For example, if you have the filtered gene list capped at 100 genes and the number of genes selected by the filtering is greater than 100, the genes are entered into the gene list in order of the filters in the sort order list.

In this example, there are 136 genes meeting the filtering criteria and we are choosing the top 10 of these based on p-value, followed by fold-change. The results of this filtering are shown in Figures 9.6 to 9.9.

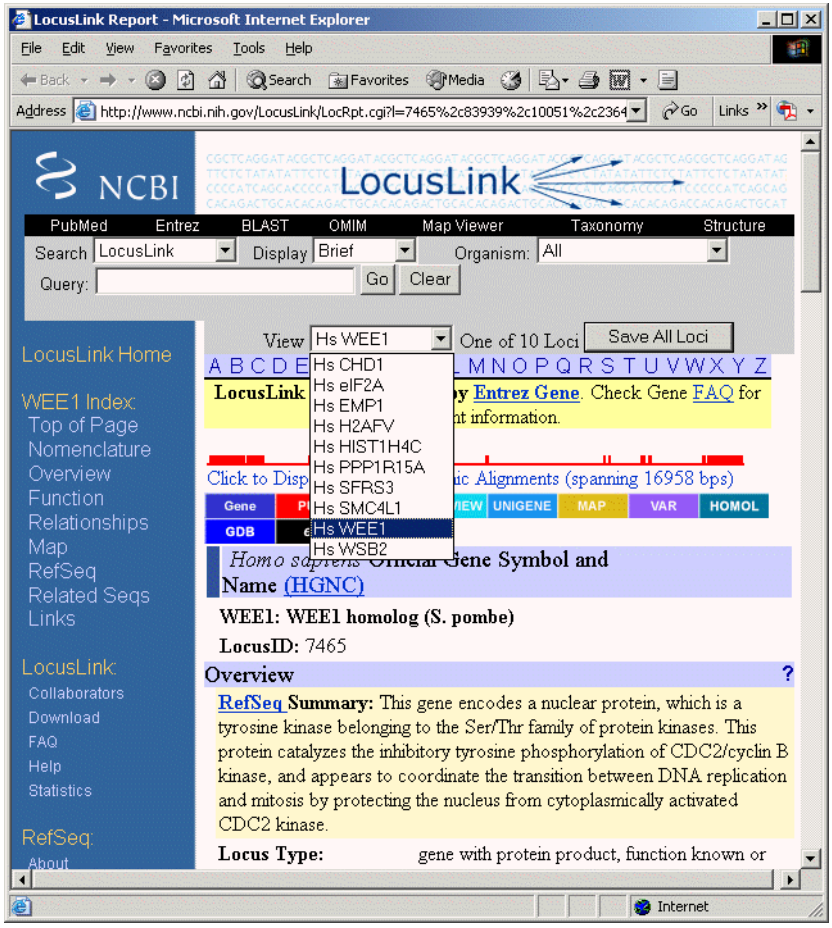


Figure 9.6: *LocusLink results for Melanoma data with fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case); top 10 genes based on p-value, followed by fold-change.*

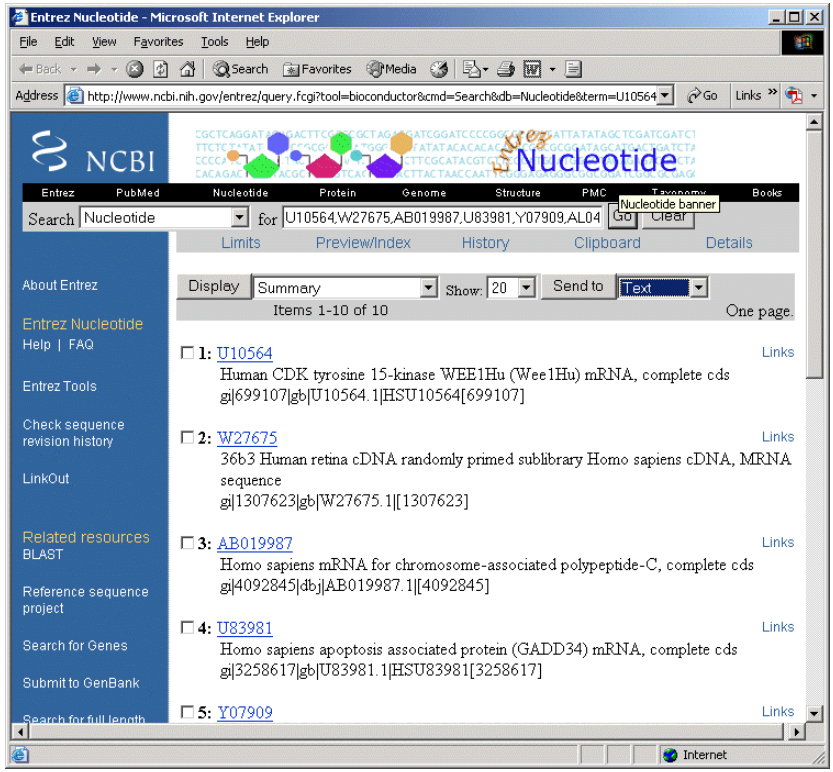


Figure 9.7: *Entrez Nucleotide results for Melanoma data with fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case); top 10 genes based on p-value, followed by fold-change.*

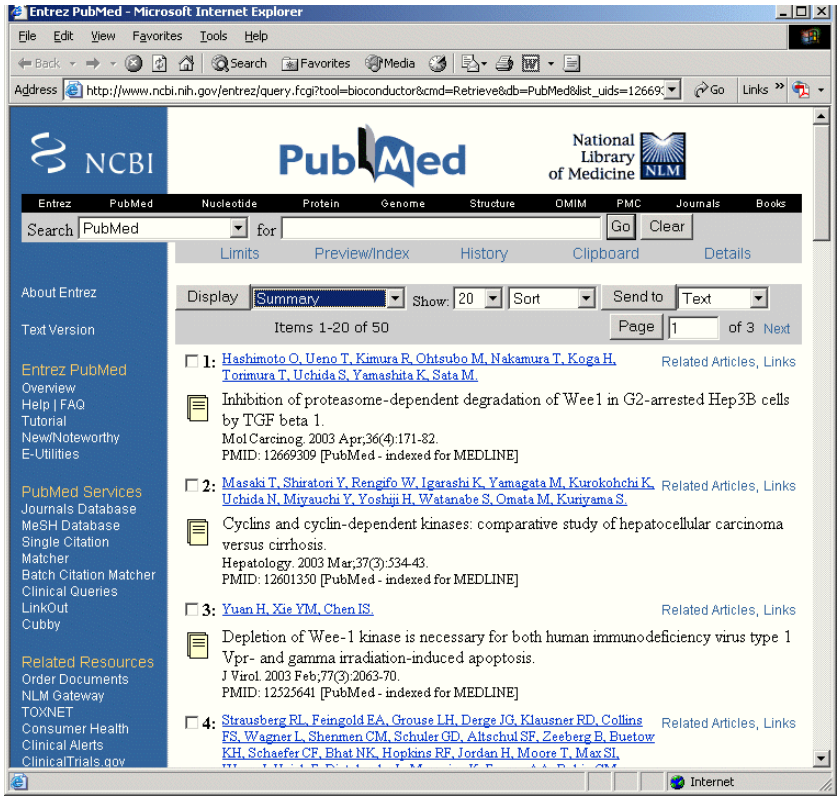

Entrez PubMed - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.ncbi.nih.gov/entrez/query.fcgi?tool=bioconductor&cmd=Retrieve&db=PubMed&list_uids=12669
NCBI
PubMed
National Library of Medicine
Entrez PubMed Nucleotide Protein Genome Structure OMIM PMC Journals Books
Search PubMed for Go Clear
Limits Preview/Index History Clipboard Details
About Entrez
Text Version
Entrez PubMed
Overview
Help | FAQ
Tutorial
New/Noteworthy
E-Utilities
PubMed Services
Journals Database
MeSH Database
Single Citation Matcher
Batch Citation Matcher
Clinical Queries
LinkOut
Cubby
Related Resources
Order Documents
NLM Gateway
TOXNET
Consumer Health
Clinical Alerts
ClinicalTrials.gov
Display Summary Show: 20 Sort Send to Text
Items 1-20 of 50
Page 1 of 3 Next
1: Hashimoto O, Ueno T, Kimura R, Ohtsubo M, Nakamura T, Koga H, Torimura T, Uchida S, Yamashita K, Sata M. Related Articles, Links
Inhibition of proteasome-dependent degradation of Wee1 in G2-arrested Hep3B cells by TGF beta 1.
Mol Carcinog. 2003 Apr;36(4):171-82.
PMID: 12669309 [PubMed - indexed for MEDLINE]
2: Masaki T, Shiratori Y, Rengifo W, Igarashi K, Yamagata M, Kurokohchi K, Uchida N, Miyauchi Y, Yoshiji H, Watanabe S, Omata M, Kuriyama S. Related Articles, Links
Cyclins and cyclin-dependent kinases: comparative study of hepatocellular carcinoma versus cirrhosis.
Hepatology. 2003 Mar;37(3):534-43.
PMID: 12601350 [PubMed - indexed for MEDLINE]
3: Yuan H, Xie YM, Chen IS. Related Articles, Links
Depletion of Wee-1 kinase is necessary for both human immunodeficiency virus type 1 Vpr- and gamma irradiation-induced apoptosis.
J Virol. 2003 Feb;77(3):2063-70.
PMID: 12525641 [PubMed - indexed for MEDLINE]
4: Strausberg RL, Feingold EA, Grouse LH, Derge JG, Klausner RD, Collins FS, Wagner L, Shenmen CM, Schuler GD, Altschul SF, Zeeberg B, Buetow KH, Schaefer CF, Bhat NK, Hopkins RF, Jordan H, Moore T, Max SI, Related Articles, Links
Internet

Figure 9.8: *Pubmed results for Melanoma data with fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case); top 10 genes based on p-value, followed by fold-change.*

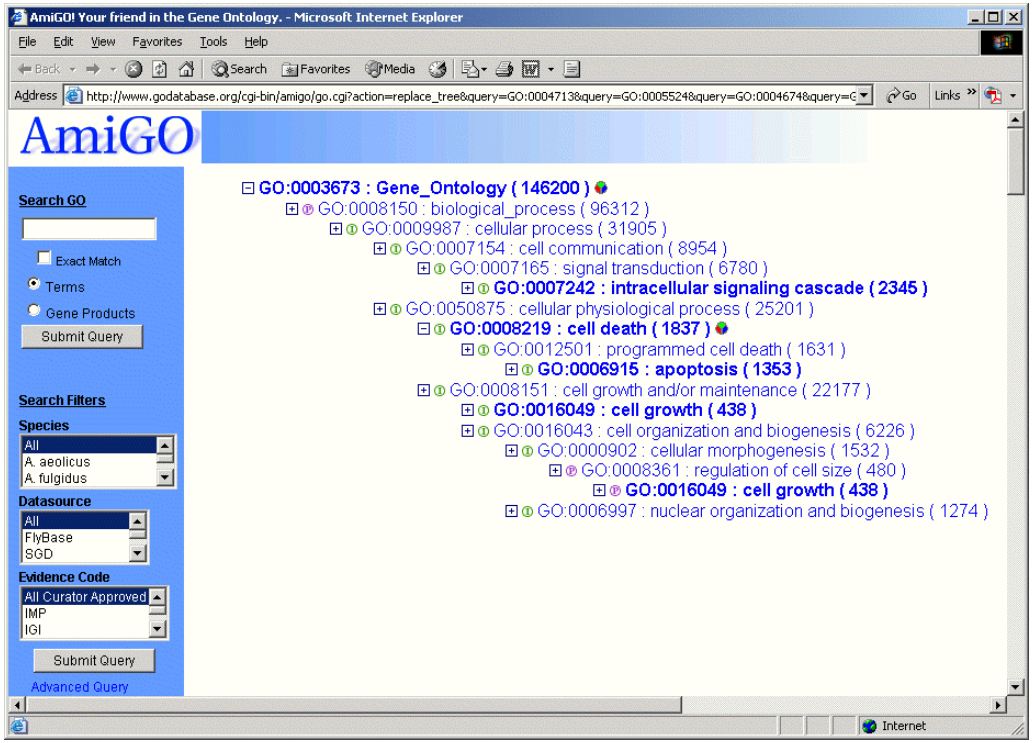

AmiGO! Your friend in the Gene Ontology. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.godatabase.org/cgi-bin/amigo/go.cgi?action=replace_tree&query=GO:0004713&query=GO:0005524&query=GO:0004674&query=G
AmiGO
Search GO
Exact Match
Terms
Gene Products
Submit Query
Search Filters
Species
All
A. aeolicus
A. fulgidus
Datasource
All
FlyBase
SGD
Evidence Code
All Curator Approved
IMP
IGI
Submit Query
Advanced Query
GO:0003673 : Gene_Ontology (146200)
GO:0008150 : biological_process (96312)
GO:0009987 : cellular process (31905)
GO:0007154 : cell communication (8954)
GO:0007165 : signal transduction (6780)
GO:0007242 : intracellular signaling cascade (2345)
GO:0050875 : cellular physiological process (25201)
GO:0008219 : cell death (1837)
GO:0012501 : programmed cell death (1631)
GO:0006915 : apoptosis (1353)
GO:0008151 : cell growth and/or maintenance (22177)
GO:0016049 : cell growth (438)
GO:0016043 : cell organization and biogenesis (6226)
GO:0000902 : cellular morphogenesis (1532)
GO:0008361 : regulation of cell size (480)
GO:0016049 : cell growth (438)
GO:0006997 : nuclear organization and biogenesis (1274)
Internet

Figure 9.9: *AmiGO results for Melanoma data with fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case); top 10 genes based on p-value, followed by fold-change.*

Annotation from the GUI: Genelist Analysis

We saw above the simple metadata lookup available for a gene list derived in S+ARRAYANALYZER with on-line databases at GenBank (LocusLink, Entrez Nucleotide, Pubmed) and the Gene Ontology Consortium (GO, AmiGO), via the **Annotation** dialog. Some other Web sites are available for statistical analysis of genelists; for example, functional gene enrichment, where we ask the question:

> *Are there more genes at a given GO node (e.g., the biological function GO tree) in a gene list than we would expect by chance?*

S+ARRAYANALYZER includes access and gene list upload to three sites for remote gene list analysis:

1. Affymetrix NetAffx

   http://www.affymetrix.com/analysis/index.affx

2. NIH DAVID/EASE

   http://apps1.niaid.nih.gov/david/

3. Onto-Express

   http://vortex.cs.wayne.edu/ontoexpress/onto.htm

These sites allow remote upload of gene list AffyIDs and/or LocusLink IDs as enabled by S+ARRAYANALYZER. We also illustrate in this section remote access and gene list upload to the Stanford Source site; information from Stanford Source is used in the Bioconductor construction of the annotation data used in S+ARRAYANALYZER.

Annotation Using Affymetrix IDs

Figure 9.10 shows the **General Options** page of the **Annotation** dialog with options chosen from **Use Affymetrix IDs** group. This writes out a file of Affymetrix IDs (**ProbeList.txt** by default) corresponding to the genes selected according to the dialog options. This **ProbeList.txt** file can be uploaded to a variety of Web sites for annotation and gene list analysis. Two options that are made easily accessible with S+ARRAYANALYZER are the Affymetrix NetAffx site and the NIH DAVID/EASE site. These two sites may be opened from the S+ARRAYANALYZER **Annotation** dialog so that all the user needs to do is browse to the **ProbeList.txt** file and upload it.

Figures 9.11 to 9.15 on the following pages show screen shots from the Affymetrix NetAffx site and the NIH DAVID/EASE site as launched and uploaded from S+ARRAYANALYZER.

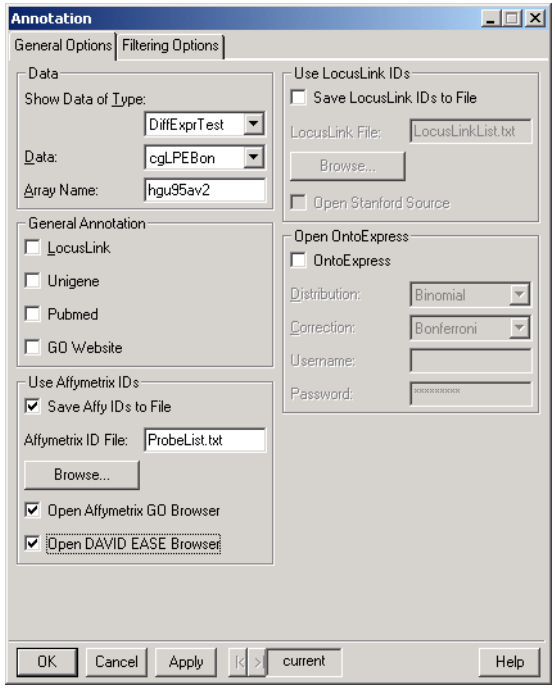


Figure 9.10: *The **General Options** page of the **Annotation** dialog. Options chosen **Use Affymetrix IDs** group. This writes out a file of Affymetrix IDs (**ProbeList.txt** by default) corresponding to the genes selected according to the **Annotation** dialog options. This **ProbeList.txt** file can be uploaded.*

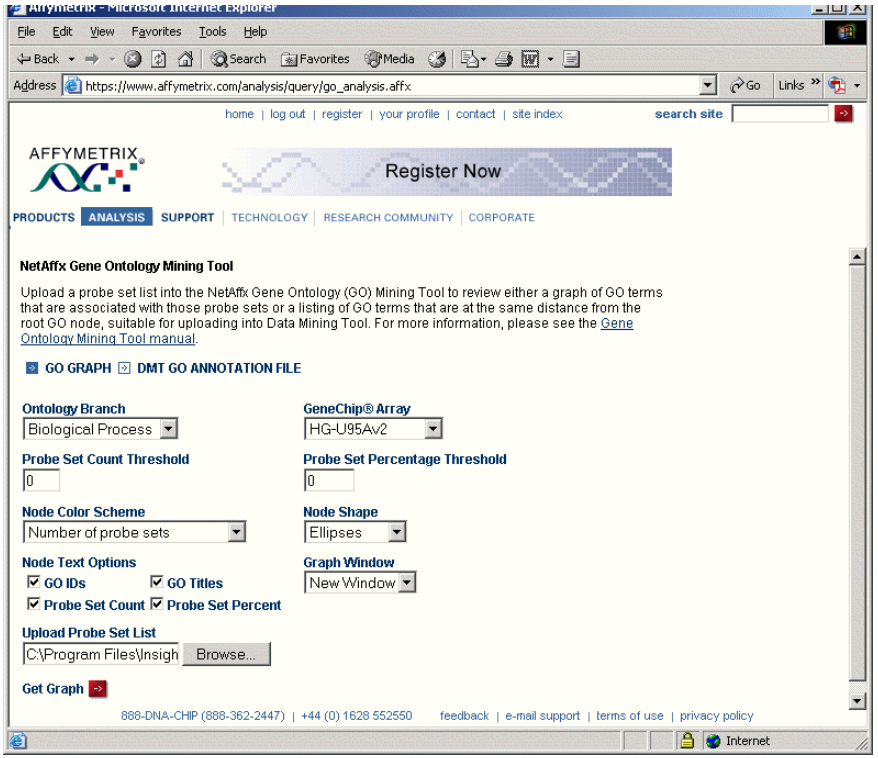

File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://www.affymetrix.com/analysis/query/go_analysis.affx
Go Links
home | log out | register | your profile | contact | site index
search site
AFFYMETRIX
Register Now
PRODUCTS ANALYSIS SUPPORT TECHNOLOGY RESEARCH COMMUNITY CORPORATE
NetAffx Gene Ontology Mining Tool
Upload a probe set list into the NetAffx Gene Ontology (GO) Mining Tool to review either a graph of GO terms that are associated with those probe sets or a listing of GO terms that are at the same distance from the root GO node, suitable for uploading into Data Mining Tool. For more information, please see the Gene Ontology Mining Tool manual.
GO GRAPH DMT GO ANNOTATION FILE
Ontology Branch
Biological Process
GeneChip® Array
HG-U95Av2
Probe Set Count Threshold
0
Probe Set Percentage Threshold
0
Node Color Scheme
Number of probe sets
Node Shape
Ellipses
Node Text Options
GO IDs
GO Titles
Probe Set Count
Probe Set Percent
Graph Window
New Window
Upload Probe Set List
C:\Program Files\Insigh
Browse...
Get Graph
888-DNA-CHIP (888-362-2447) | +44 (0) 1628 552550
feedback | e-mail support | terms of use | privacy policy
Internet

Figure 9.11: *Affymetrix NetAffx Web site, launched by S+ARRAYANALYZER with list of IDs for Melanoma data.*

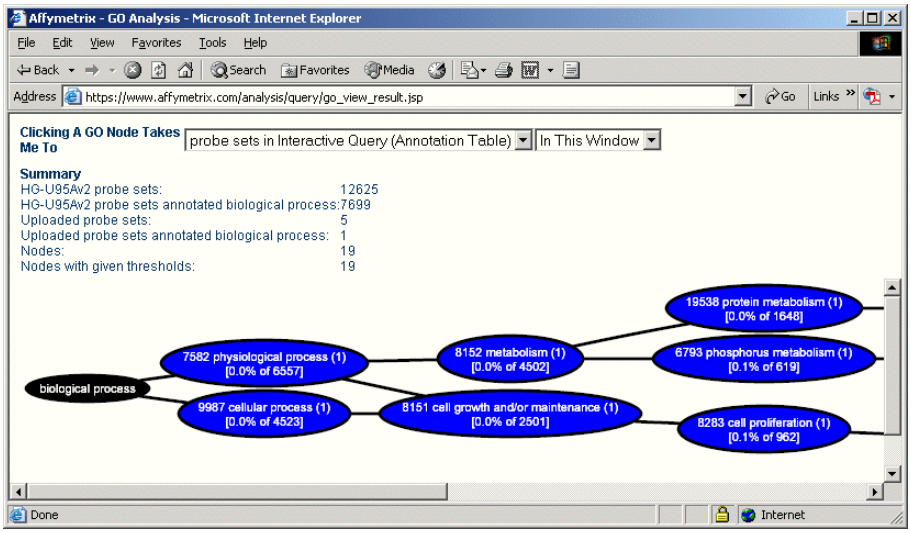

Affymetrix - GO Analysis - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://www.affymetrix.com/analysis/query/go_view_result.jsp
Go Links
Clicking A GO Node Takes Me To
probe sets in Interactive Query (Annotation Table)
In This Window
Summary
HG-U95Av2 probe sets: 12625
HG-U95Av2 probe sets annotated biological process: 7699
Uploaded probe sets: 5
Uploaded probe sets annotated biological process: 1
Nodes: 19
Nodes with given thresholds: 19
19538 protein metabolism (1) [0.0% of 1648]
7582 physiological process (1) [0.0% of 6557]
8152 metabolism (1) [0.0% of 4502]
6793 phosphorus metabolism (1) [0.1% of 619]
biological process
9987 cellular process (1) [0.0% of 4523]
8151 cell growth and/or maintenance (1) [0.0% of 2501]
8283 cell proliferation (1) [0.1% of 962]
Done
Internet

Figure 9.12: *Affymetrix NetAffx GO browser with list of IDs for Melanoma data with fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case); top 10 genes based on p-value, followed by fold-change.*

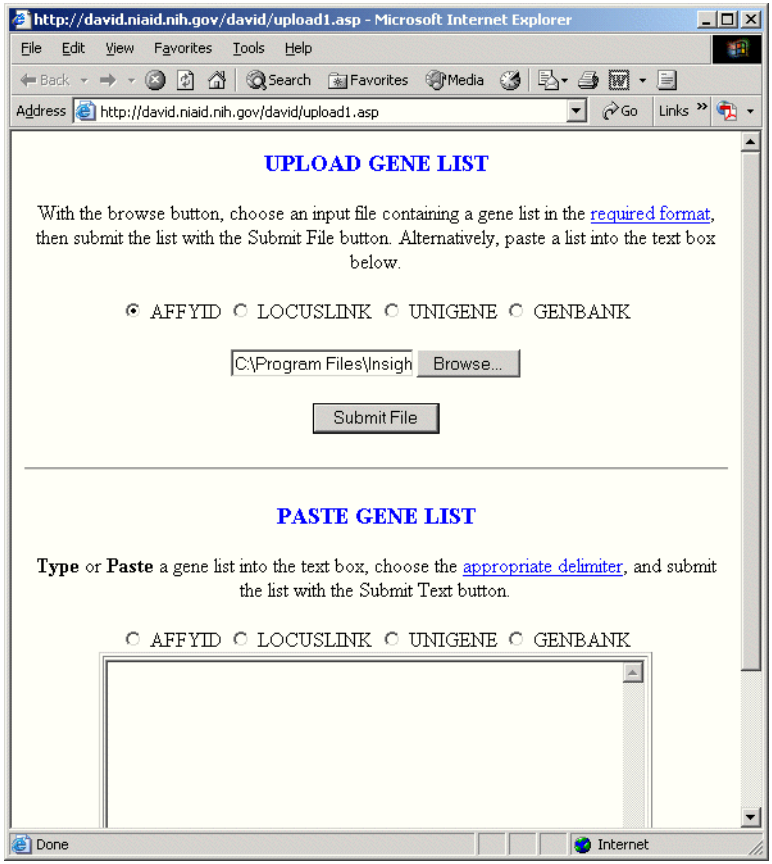

http://david.niaid.nih.gov/david/upload1.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://david.niaid.nih.gov/david/upload1.asp
Go Links
UPLOAD GENE LIST
With the browse button, choose an input file containing a gene list in the required format, then submit the list with the Submit File button. Alternatively, paste a list into the text box below.
AFFYID LOCUSLINK UNIGENE GENBANK
C:\Program Files\Insigh
Browse...
Submit File
PASTE GENE LIST
Type or Paste a gene list into the text box, choose the appropriate delimiter, and submit the list with the Submit Text button.
AFFYID LOCUSLINK UNIGENE GENBANK
Done
Internet

Figure 9.13: *DAVID/EASE Web site, launched by S+ARRAYANALYZER with list of IDs for Melanoma data.*

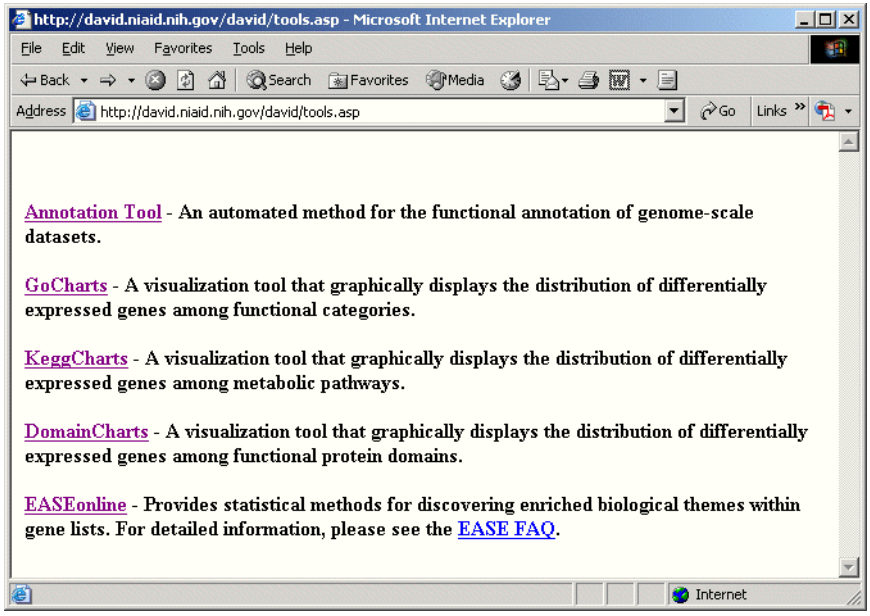
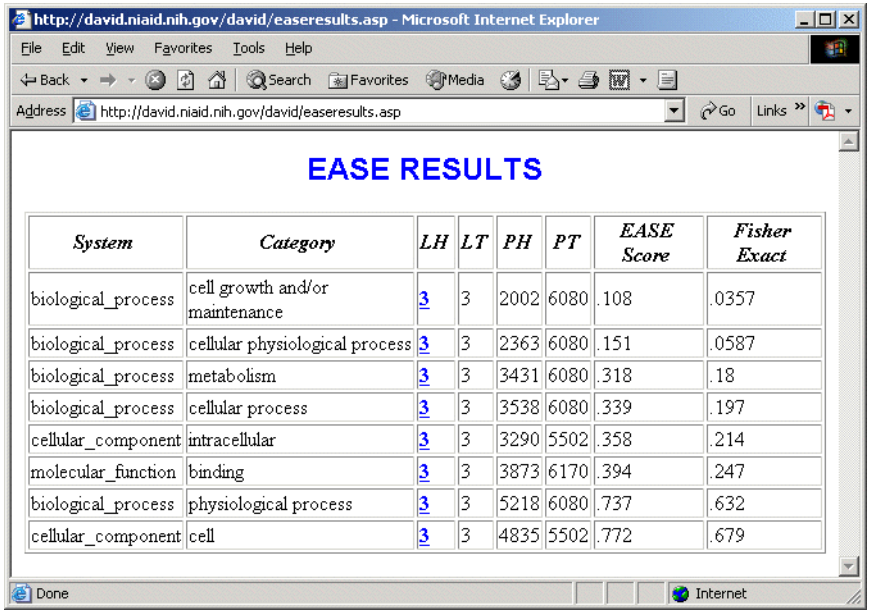

http://david.niaid.nih.gov/david/tools.asp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://david.niaid.nih.gov/david/tools.asp
Go Links
Annotation Tool - An automated method for the functional annotation of genome-scale datasets.
GoCharts - A visualization tool that graphically displays the distribution of differentially expressed genes among functional categories.
KeggCharts - A visualization tool that graphically displays the distribution of differentially expressed genes among metabolic pathways.
DomainCharts - A visualization tool that graphically displays the distribution of differentially expressed genes among functional protein domains.
EASEonline - Provides statistical methods for discovering enriched biological themes within gene lists. For detailed information, please see the EASE FAQ.
Internet

Figure 9.14: *DAVID/EASE Web site, analysis selections*



System	*Category*	*LH*	*LT*	*PH*	*PT*	*EASE Score*	*Fisher Exact*
biological_process	cell growth and/or maintenance	3	3	2002	6080	.108	.0357
biological_process	cellular physiological process	3	3	2363	6080	.151	.0587
biological_process	metabolism	3	3	3431	6080	.318	.18
biological_process	cellular process	3	3	3538	6080	.339	.197
cellular_component	intracellular	3	3	3290	5502	.358	.214
molecular_function	binding	3	3	3873	6170	.394	.247
biological_process	physiological process	3	3	5218	6080	.737	.632
cellular_component	cell	3	3	4835	5502	.772	.679

Figure 9.15: *DAVID/EASE site, Fisher exact test for gene enrichment for list of Melanoma data Affy IDs uploaded by S+ARRAYANALYZER.*

Annotation Using LocusLink IDs

Figure 9.16 shows the **General Options** page of the **Annotation** dialog with options chosen from **Use LocusLink IDs** group. This writes out a file of LocusLink IDs (**LocusLinkList.txt** by default) corresponding to the genes selected according to the **Annotation** dialog options. This **LocusLinkList.txt** file can be uploaded.

One option that is made easily accessible with S+ArrayAnalyzer is the Stanford Source site. This site may be opened from the **Annotation** dialog so that all the user needs to do is browse to the file **LocusLinkList.txt** and upload it. Figure 9.17 shows a screen shot from the Stanford Source site as launched and uploaded from S+ARRAYANALYZER. Information from Stanford Source is used in constructing the annotation data in Bioconductor.

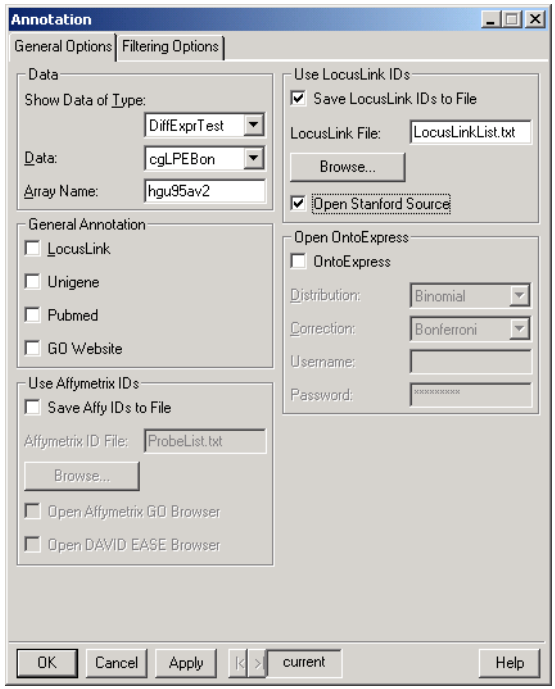


Figure 9.16: *The **General Options** page of the **Annotation** dialog. Options chosen **Use LocuLink IDs** group. This writes out a file of LocusLink IDs (**LocusLinkList.txt** by default) corresponding to the genes selected according to the **Annotation** dialog options.*

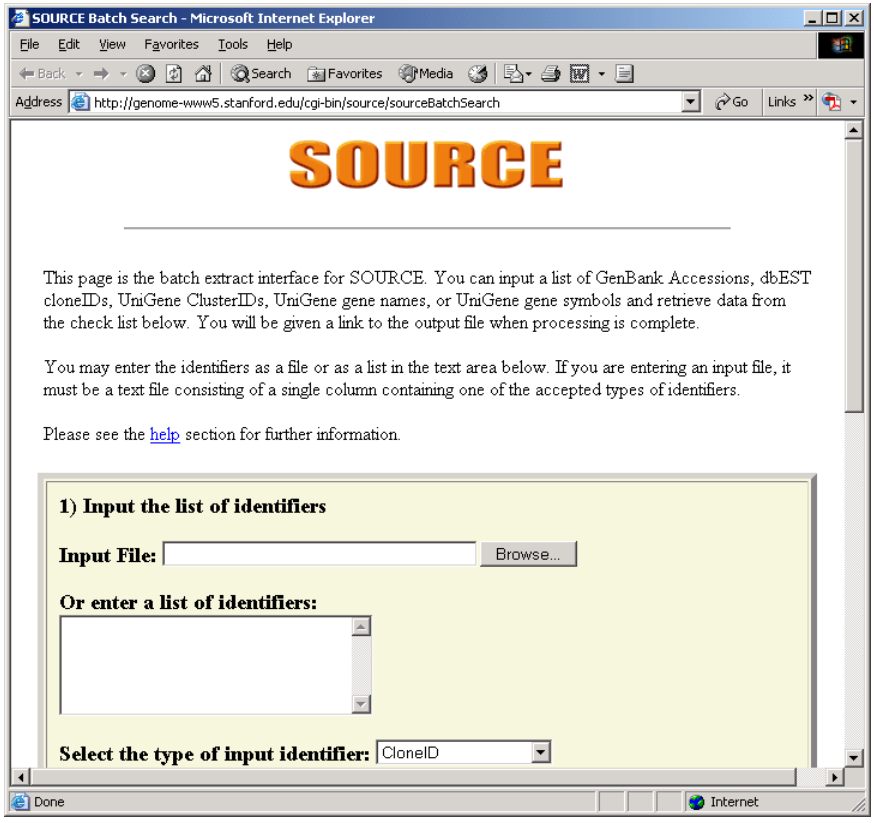

SOURCE Batch Search - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://genome-www5.stanford.edu/cgi-bin/source/sourceBatchSearch
Go Links
SOURCE
This page is the batch extract interface for SOURCE. You can input a list of GenBank Accessions, dbEST cloneIDs, UniGene ClusterIDs, UniGene gene names, or UniGene gene symbols and retrieve data from the check list below. You will be given a link to the output file when processing is complete.
You may enter the identifiers as a file or as a list in the text area below. If you are entering an input file, it must be a text file consisting of a single column containing one of the accepted types of identifiers.
Please see the help section for further information.
1) Input the list of identifiers
Input File:
Browse...
Or enter a list of identifiers:
Select the type of input identifier:
CloneID
Done
Internet

Figure 9.17: *Stanford Source site launched from S+ARRAYANALYZER with LocusLink gene list upload*

Annotation Using OntoExpress

Figure 9.18 shows the **General Options** page of the **Annotation** dialog with options chosen from **Open OntoExpress** group. This sends a log-in message to the Onto-Express site:

http://vortex.cs.wayne.edu/projects.htm#Onto-Express

You will need an account on OntoExpress for this log-in to be successful. You can select options for a gene enrichment analysis on OntoExpress from S+ARRAYANALYZER including the distribution of the test (binomial, hypergeometric and chi-square) and FWER/FDR correction (Bonferroni, Sidak, Holm, BH-FRD).

S+ARRAYANALYZER sends the selected, filtered gene list of IDs to OntoExpress for this analysis. OntoExpress then opens itself in a java applet with the analyses done and additional analyses available. In this case, we send the Melanoma data with fold-change set to greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case); top 10 genes based on P-Value, followed by fold-change. Results are displayed in the OntoExpress applet in Figure 9.19.

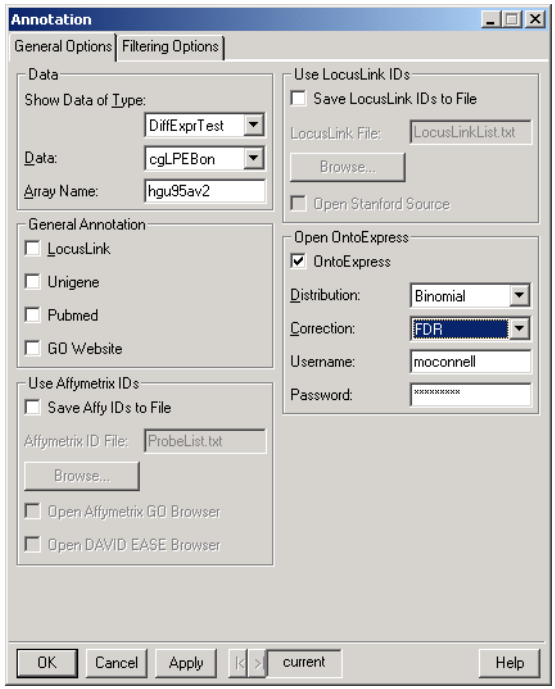


Figure 9.18: *The **General Options** page of the **Annotation** dialog. Options chosen are from the **Open OntoExpress** group*

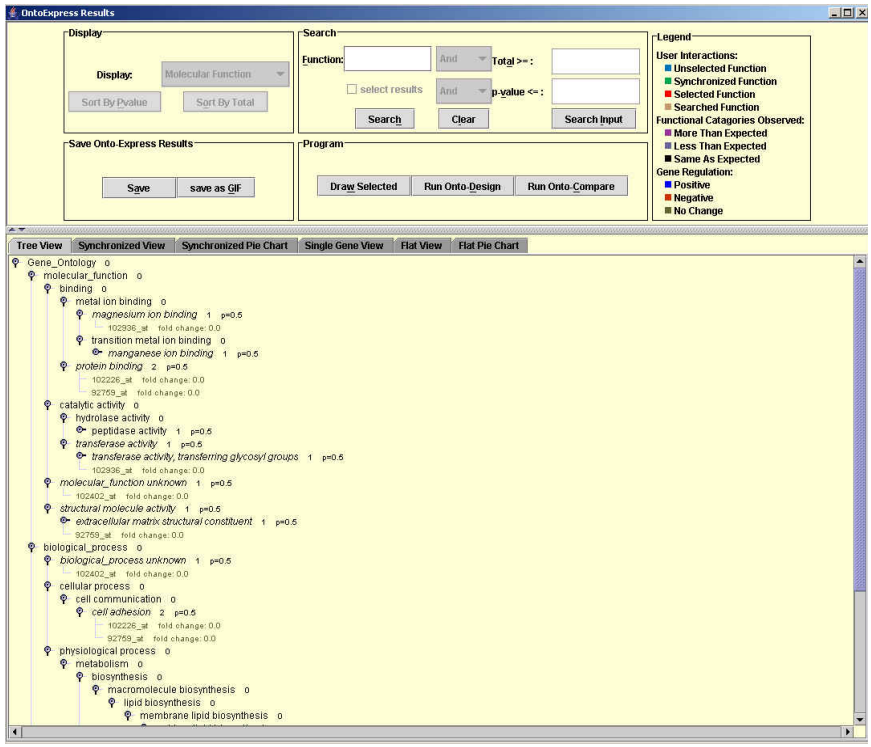


Figure 9.19: *OntoExpress java applet as launched from S+ARRAYANALYZER showing tests for functional enrichment for the uploaded gene list using settings chosen from within S+ARRAYANALYZER.*

Annotation from Command Line Scripting: GenBank Metadata Lookups

We saw above the simple metadata lookup available for a gene list derived from statistical analyses using the in the **Annotation** dialog. We now show similar functionality from the S+ARRAYANALYZER command line scripting environment

```
## Load libraries for annotation
> module(arrayanalyzer)
> library(hgu95av2AnnoData)

## 1. Setup the data for analysis
## Make a copy of the Melanoma S-PLUS object
## from RMA and LPE with Bonferroni FWER correction
summObj <- cgLPEBon@allData
```

```
## 2. Filter the data to genes with fold change
## greater than 2 and LPE p-value less than 0.001
fc2.p001 <- summObj[summObj$AdjPvalue < 0.001 &
summObj$foldChange > 2, ]
mel.gnames <- fc2.p001$GeneName

## 3. Get LocusLink IDs and make a call to locuslink
mel.llnames <-
as.numeric(unlist(hgu95av2LOCUSID[mel.gnames]))
locuslinkByID(mel.llnames)

## 4. Get accession numbers and call to Entrez
uids <- unlist(hgu95av2ACCNUM[fc2.p001$GeneName])
genbank(uids, type="accession",disp="browser")

## 5. Get Pubmed ids and call to Pubmed for articles
pmedids <- hgu95av2PMID[mel.gnames]
pubmed(pmedids, disp="browser")
```

In this example, the filtering of the Melanoma data with fold-change greater than 2 and p-value set to significant (adjusted $p < 0.05$ for LPE test, Bonferroni FWER in this case) produces four genes. The LocusLink information for these four genes is obtained using the S-PLUS function `locuslinkByID`. This returns the Web page shown in Figure 9.20. Note that the **View** field in LocusLink is populated with the four genes identified in the analysis.

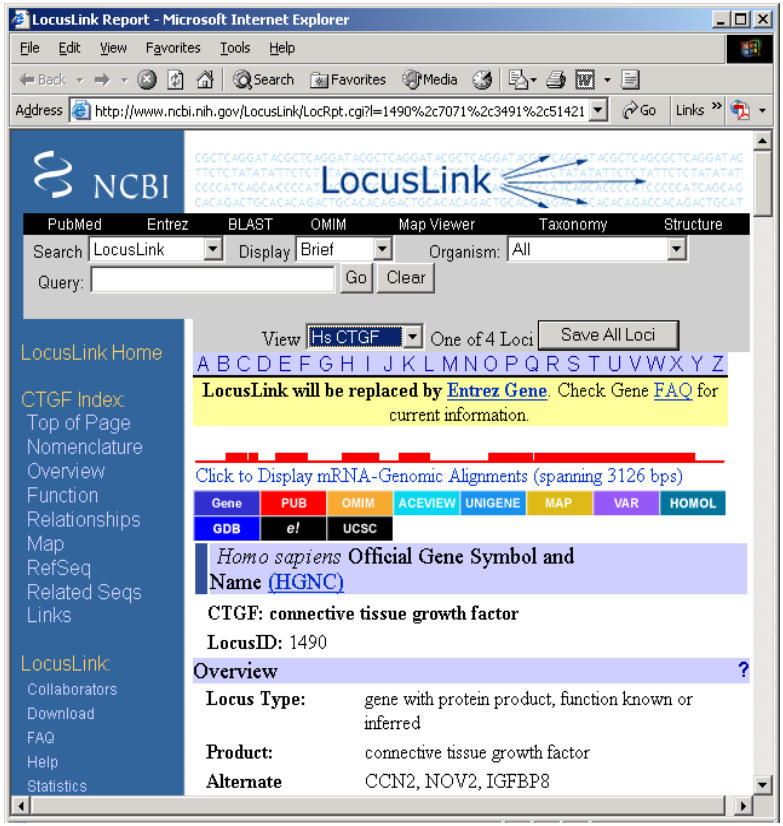

LocusLink Report - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.ncbi.nih.gov/LocusLink/LocRpt.cgi?l=1490%2c7071%2c3491%2c51421
NCBI
LocusLink
PubMed Entrez BLAST OMIM Map Viewer Taxonomy Structure
Search LocusLink Display Brief Organism: All
Query: Go Clear
LocusLink Home
CTGF Index:
Top of Page
Nomenclature
Overview
Function
Relationships
Map
RefSeq
Related Seqs
Links
LocusLink:
Collaborators
Download
FAQ
Help
Statistics
View Hs.CTGF One of 4 Loci Save All Loci
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
LocusLink will be replaced by Entrez Gene. Check Gene FAQ for current information.
Click to Display mRNA-Genomic Alignments (spanning 3126 bps)
Gene PUB OMIM ACEVIEW UNIGENE MAP VAR HOMOL
GDB e! UCSC
Homo sapiens Official Gene Symbol and Name (HGNC)
CTGF: connective tissue growth factor
LocusID: 1490
Overview
Locus Type: gene with protein product, function known or inferred
Product: connective tissue growth factor
Alternate CCN2, NOV2, IGFBP8

Figure 9.20: *LocusLink annotation for the four genes identified in the gene filtering analysis described above. Note that the* ***View*** *field in LocusLink is populated with the four genes identified in the analysis.*

There are several other simple metadata lookup queries we can run from the command line scripting environment for the gene list that we are holding in the S-PLUS variable `mel.gnames`. We illustrate by obtaining accession number information and Pubmed articles. S-PLUS code for this annotation follows, and results from the queries are presented in Figures 9.21 and 9.22.

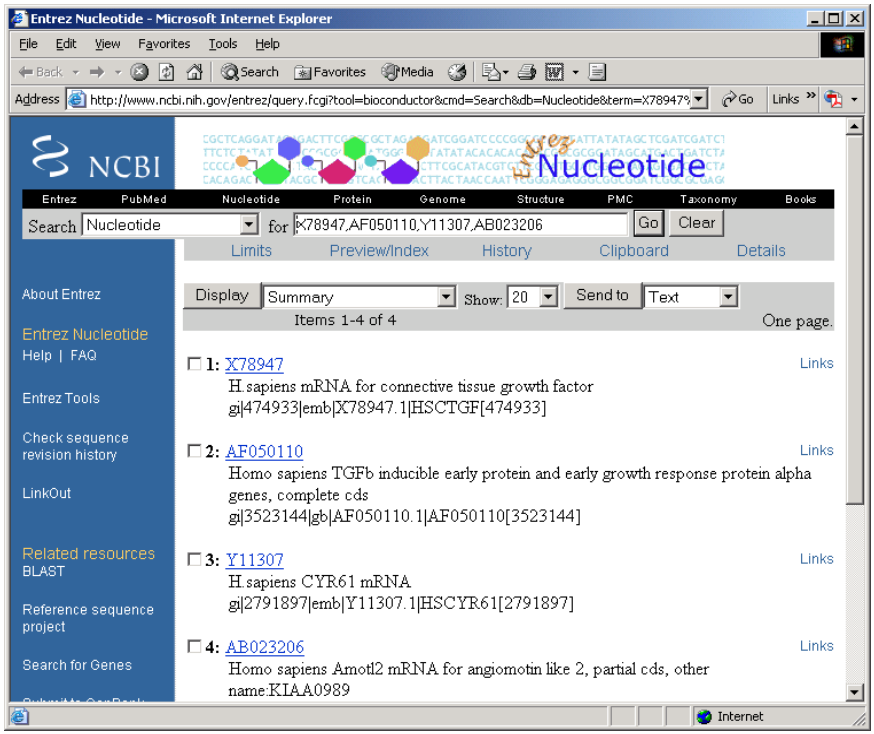


Figure 9.21: *Accession number lookup in Entrez for the four genes identified in the gene filtering analysis described above.*

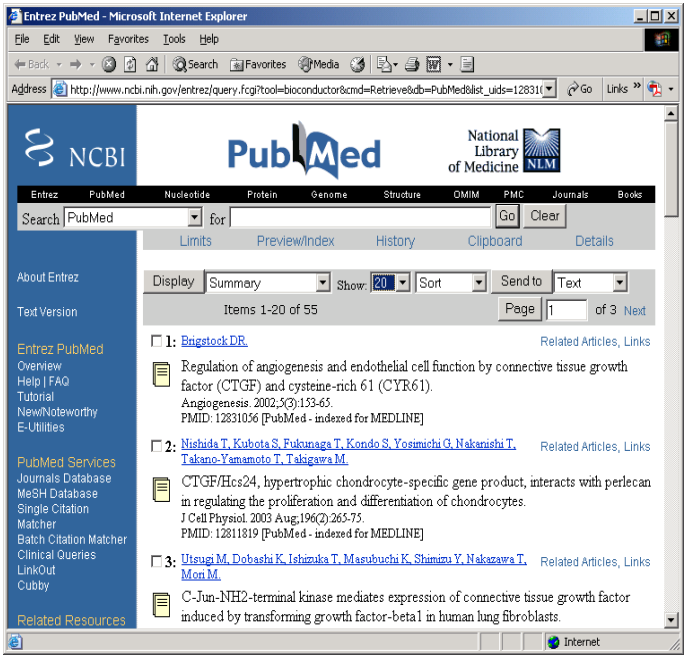

Entrez PubMed - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.ncbi.nih.gov/entrez/query.fcgi?tool=bioconductor&cmd=Retrieve&db=PubMed&list_uids=12831
NCBI
PubMed
National Library of Medicine
Entrez PubMed Nucleotide Protein Genome Structure OMIM PMC Journals Books
Search PubMed for Go Clear
Limits Preview/Index History Clipboard Details
About Entrez
Text Version
Entrez PubMed
Overview
Help | FAQ
Tutorial
New/Noteworthy
E-Utilities
PubMed Services
Journals Database
MeSH Database
Single Citation Matcher
Batch Citation Matcher
Clinical Queries
LinkOut
Cubby
Related Resources
Display Summary Show: 20 Sort Send to Text
Items 1-20 of 55
Page 1 of 3 Next
1: Brigstock DR. Related Articles, Links
Regulation of angiogenesis and endothelial cell function by connective tissue growth factor (CTGF) and cysteine-rich 61 (CYR61).
Angiogenesis. 2002;5(3):153-65.
PMID: 12831056 [PubMed - indexed for MEDLINE]
2: Nishida T, Kubota S, Fukunaga T, Kondo S, Yosimichi G, Nakanishi T, Takano-Yamamoto T, Takigawa M. Related Articles, Links
CTGF/Hcs24, hypertrophic chondrocyte-specific gene product, interacts with perlecan in regulating the proliferation and differentiation of chondrocytes.
J Cell Physiol. 2003 Aug;196(2):265-75.
PMID: 12811819 [PubMed - indexed for MEDLINE]
3: Utsugi M, Dobashi K, Ishizuka T, Masubuchi K, Shimizu Y, Nakazawa T, Mori M. Related Articles, Links
C-Jun-NH2-terminal kinase mediates expression of connective tissue growth factor induced by transforming growth factor-beta1 in human lung fibroblasts.
Internet

Figure 9.22: *Pubmed articles for the four genes identified in the gene filtering analysis described above.*

Filtering Genes Based on GO Categories

GO categories can be used to filter data to particular gene subsets of interest before differential expression testing and/or other analyses. For example, in a study involving neutrophil production, one may be specifically interested in expression of genes associated with the GO term 'defense response'. In the case of such targeted interest, restriction to these pathways *a priori* can have considerable advantages compared to the approach of considering all genes simultaneously. In particular, FWER/FDR adjustments are more contained, and smaller (but real) effects can be detected more often.

The GO filtering process includes the following four components:

1. Select the GO term(s) of interest.
2. Identify the set of assayed genes that are annotated at that GO term. Identifiers can be, for example, the Affy IDs for genes on the chip with the GO annotation of interest.
3. Subset the expression data set using the appropriate identifiers.

4. Perform the analysis (e.g., differential expression) on the expression data subset.

We illustrate this *a priori* filtering pipeline with a study on the progression of granulocyte differentiation. The study was done using Affymetrix mgu74av2 arrays as a time course experiment in which a model cell line was used to elucidate mechanisms by which retinoic acid signaling causes promelocytes to stop dividing and turn into mature neutrophils. The progression under study is shown below in Figure 9.23. A more detailed explanation and analysis of these data is available by e-mailing a request to

S-PLUS.pharma@insightful.com)

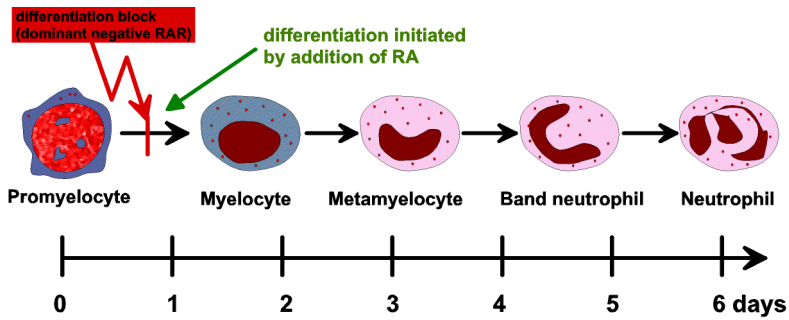


Figure 9.23: *Study of granulocyte differentiation.*

In studies such as this, there may be *a priori* interest in a particular pathway or cell response ontology. For illustrative purposes only, we choose to focus our attention on the "defense response" Biological Process ontology for our filtering and hypothesis testing exercise. We may wish to view the GO hierarchy for defense response e.g. by using the NetAffx GO browser on the Affymetrix web site. This provides a contextual view of the biological processes within this GO node. For more information, see Figure 9.24 below.

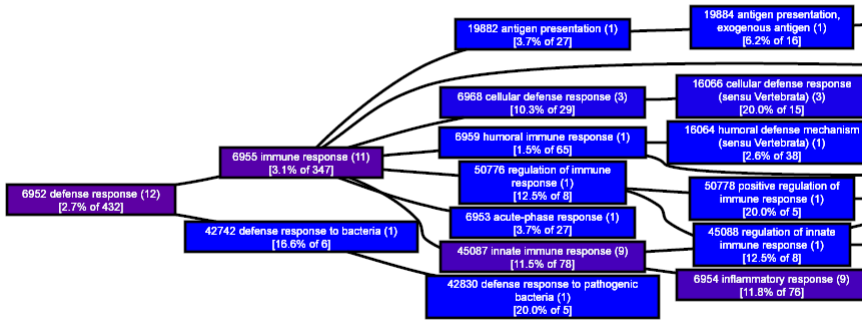

19882 antigen presentation (1)
[3.7% of 27]
19884 antigen presentation, exogenous antigen (1)
[5.2% of 16]
6968 cellular defense response (3)
[10.3% of 29]
16066 cellular defense response (sensu Vertebrata) (3)
[20.0% of 15]
6959 humoral immune response (1)
[1.5% of 65]
16064 humoral defense mechanism (sensu Vertebrata) (1)
[2.6% of 38]
6955 immune response (11)
[3.1% of 347]
50776 regulation of immune response (1)
[12.5% of 8]
6952 defense response (12)
[2.7% of 432]
50778 positive regulation of immune response (1)
[20.0% of 5]
6953 acute-phase response (1)
[3.7% of 27]
42742 defense response to bacteria (1)
[16.6% of 6]
45088 regulation of innate immune response (1)
[12.5% of 8]
45087 innate immune response (9)
[11.5% of 78]
6954 inflammatory response (9)
[11.8% of 76]
42830 defense response to pathogenic bacteria (1)
[20.0% of 5]

Figure 9.24: *Defense response hierarchy in GO Biological Process ontology*

All that S+ARRAYANALYZER needs to do in the filtering is the GO term. The GO term can be selected from the GOAnnoData library (Step 1, below). This GO term is then used to pick off the corresponding Affy IDs on the mgu74av2 chip (Step 2, below). These Affy IDs are then used to subset the S+ARRAYANALYZER data set prior to differential expression testing (Step 3, below).

```
## 1. Select GO terms of interest
## Load the GO AnnoData library (if not already loaded)
goterms <- unlist(GOBPID2TERM)
position <- match("defense response", goterms)
def.resp <- names(GOBPID2TERM)[position]
def.resp
[1] "GO:0006952"

## 2. Obtain identifiers (Affy IDs) for genes on the chip
## with the GO annotation of interest
> library(mgu74av2AnnoData)
> def.IDs <-
mgu74av2GO2ALLPROBES[names(mgu74av2GO2ALLPROBES)==def.resp]

## 3. Subset the expression dataset to genes
## with the GO annotation of interest
```

```
> DAYSdefense <- DAYSrmaExprSet
> exprs(DAYSdefense)<-exprs(DAYSrmaExprSet.summ)[def.IDs,]

## 4. Perform differential expression analysis on the
## expression data subset
```

The filtering operation in Step 3 above reduces the number of genes from 12,488 on the mgu74av2 chip to 540. This allows more focused differential expression testing. Note that we are filtering only the exprs slot i.e. the actual expression data. All other slots remain untouched. There are several simple ways to do this. In the above we start with the full data set `DAYSrmaExprSet`. This data set represents all 7 time points (days) and 4 chips per time point, with the 28 cel files summarized using RMA. We make a copy of this object and then subset the exprs slot using the Affy IDs we obtained in Steps 1 and 2. The resulting object `DAYSdefense` includes only the 540 genes in the defense response branch of the GO Biological Process ontology.

The new filtered object is available through the S+ArrayAnalyzer GUI for analysis (e.g., differential expression and clustering). A volcano plot for one of the contrasts is shown in Figure 9.25.

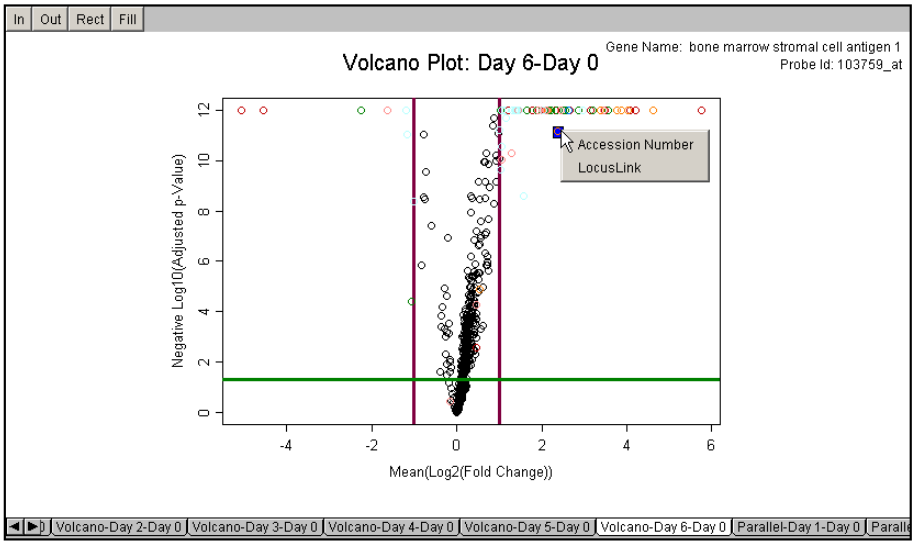


Figure 9.25: *HTML volcano plot for the day0 versus day6 baseline contrast for the* `DAYSdefense` *object, filtered by the GO Biological Process branch, "defense response." The complete ANOVA output for this analysis includes a summary page and parallel coordinates and volcano plots for each contrast.*

APPENDIX A: CREATING A DESIGN FILE

Introduction **370**

Format Specification **371**
- Special Keys 371
- ImportInfo valuenames 375
- FactorInfo valuenames 377
- Design valuenames 378
- DesignType valuenames 380
- Example Design File for Affymetrix MAS/CEL 382
- Example Design File for Two-Color Experiments 383

INTRODUCTION

Design files are used to describe the experimental conditions of a microarray analysis. Typically, you list the number of arrays to be read, the number of factors in the experiment (one or two are currently allowed), and the name, number of levels and level values for each factor.

In S+ARRAYANALYZER, you can enter your data files in the (**Import Data**) **From Affymetrix** or (**Import Data**) **From Two-Color** dialogs and associate the files with a particular name, level, and factor, but this may be inefficient if you have a large number of files or you plan on analyzing a number of experiments with a limited number of changes in each. Creating a design file to import your data files is often simpler, and you can create this design file in your favorite text editor and click the **Read Existing Design** button to read the design file.

Design files written by S+ARRAYANALYZER are always clear text files, and record specific information from the first page of the data import dialogs. The information is written between special keys that mark the beginning and end of blocks that contain data related to a specific area of the dialog. Most of these blocks are optional; however, a block, if present, must have a **START** and an **END** key.

A specific format is required for S+ARRAYANALYZER to identify and load the data from the file. The following documentation can be used to decode the meaning of the keys and to help decipher the required format.

FORMAT SPECIFICATION

The following describes a specification for the format of S+ARRAYANALYZER 2.0x design files along with rules that must be followed when writing out or modifying these files.

Design files may have any extension but must be clear text files. All the lines are read by S+ARRAYANALYZER at one time. These lines are then parsed to find keys and data blocks containing data used to populate the first import dialog page. Blank lines are ignored.

The design file must begin with a line containing the S+ARRAYANALYZER version key:

;V2.01

This version key changes from version to version, and represents the version used for the file. The current version is 2.01.

If this line is not found, the file is interpreted as a version 1.1 or earlier file, for backward compatibility.

Special Keys

Note that any line beginning with a semi-colon is interpreted as a special key. Whenever a special key is found, it is expected that the first occurrence contains the flag **[START]** and that there will be another line following this line which contains the same key but with the **[END]** flag. These special keys define data blocks by the corresponding **START** and **END** lines. As an example:

;ImportInfo[START]

FileType=MAS *5* Summary Data

ChipName=mgu74av2

CDFPath=

SaveAs=MouseSwimExprSet

PrintOutput=1

;ImportInfo[END]

In this example, the special key is **ImportInfo**. The start of the **ImportInfo** block is defined by the line containing **;ImportInfo[START]** and the end by **;ImportInfo[END]**.

Any blank lines in between these keys are ignored, and any lines that are not blank are interpreted as settings. Any line in the block must follow the format:

Valuename=Value

where **Valuename** is the name of the value to set, and **Value** is the value to use. Any valuename that is not recognized is ignored. Use the tables below to determine the recognized value names. You can assign an empty value to a valuename by leaving it blank:

Valuename=

Table J.3 contains descriptions, rules, and examples of the special keys. This is followed by a set of tables, one for each special key that lists the allowable valuenames for the key along with any rules and examples.

Table J.3: *Table of special keys.*

Line	Description	Rules	Example
;V2.01	Version key: Identifies the version of S+ARRAYANALYZER used when the file was written.	REQUIRED. This line must be the first non-empty line of the file.	**;V2.01**
;ImportInfo[START] **…** **;ImportInfo[END]**	Import key: Identifies a block which contains information about the data imported, such as file type, array name, etc.	OPTIONAL. All lines between these two keys are interpreted as settings for the imported data. See ImportInfo valuenames table below for more information about these settings.	**;ImportInfo[START]** **FileType=MAS *5* Summary Data** **ChipName=mgu74av2** **CDFPath=** **SaveAs=MouseSwimExprSet** **PrintOutput=1** **;ImportInfo[END]**

Table J.3: *Table of special keys. (Continued)*

Line	Description	Rules	Example
;FactorInfo[START] **…** **;FactorInfo[END]**	Factor key: Identifies a block which contains information about the factors and levels used.	OPTIONAL. The lines in between these keys define the factors and levels to be used. See the **FactorInfo** value names table below for more information.	**;FactorInfo[START]** **A|ZebraFish|No|Swirl WildType** **;FactorInfo[END]**
;Design[START] **…** **;Design[END]**	Design key: Identifies a block which defines the files to import and the factors and levels to use.	OPTIONAL. The lines in between these keys define the files to import and the factors and levels to use for each file. See the **Design** valuenames table below for more information.	**;Design[START]** **relativepath= C:\Program Files\ Insightful\splus62\ module\ArrayAnalyzer\ examples** **swirl.1.spot,swirl.1,Swirl, WildType** **swirl.2.spot,swirl.2,Swirl, WildType** **swirl.3.spot,swirl.3, WildType,Swirl** **swirl.4.spot,swirl.4, WildType,Swirl** **;Design[END]**

Table J.3: *Table of special keys. (Continued)*

Line	Description	Rules	Example
;DesignType[START] **…** **;DesignType[END]**	DesignType key: Identifies a block used for Two Channel imports only which defines more design parameters as displayed in the Create/Modify Design dialog for Two Channel designs.	OPTIONAL. Only used by Two Channel imports. The lines in between these keys define additional design parameters. See the DesignType valuenames table below for more information.	**;DesignType[START]** **Type=0** **TWO_SAMPLE_ DyeSwap=1** **LOOP_Factor= ZebraFish** **LOOP_DyeSwap=1** **REFERENCE_Factor= ZebraFish** **REFERENCE_RefLevel =Swirl** **REFERENCE_RefType =2** **;DesignType[END]**

ImportInfo valuenames

Table J.4: *Table of **ImportInfo** valuenames*

Value Name	Rules	Example
FileType	OPTIONAL. Used only by Affymetrix import. If not specified, file type is auto determined by first file read. Specifies the file type for the files in the Design data block (see Design Lines table below). The file type must be one of the following recognized types: MAS 5 Summary Data MAS 4 Summary Data MAS Summary Data Probe Level (.CEL) Affymetrix CHP	FileType=MAS 5 Summary Data
ChipName	REQUIRED. For Affymetrix imports, used as the Array Name. For Two-Channel import, used as the layout object name. In this case, the name specified must be the name of an existing layout object previously created using the Create Layout dialog in ArrayAnalyzer.	ChipName=mgu74av2

Table J.4: *Table of **ImportInfo** valuenames (Continued)*

Value Name	Rules	Example
CDFPath	OPTIONAL. Only required for Affymetrix imports where FileType is Affymetrix CHP. In this case, the full pathname to the CDF file is specified. Not required for Two Channel imports.	CDFPath=d:\mycdfs\cdf1.cdf
SaveAs	OPTIONAL. Specify the name of an S object to save the imported data. WARNING: If you specify the name of an existing S object, it will be overwritten.	SaveAs=MouseSwimExprSet

Table J.4: *Table of **ImportInfo** valuenames (Continued)*

Value Name	Rules	Example
PrintOutput	OPTIONAL. If not specified, no output is printed. Specify whether to display information about the import in a report window in S-PLUS. You may specify any of the following values for this valuename to indicate true: Y Yes T True 1 all other values are interpreted as a false value	PrintOutput=1

FactorInfo valuenames

The lines in a **FactorInfo** block are different than other data blocks. There are no valuenames. Instead, each non-empty line is interpreted as a factor. Currently, only two factors are allowed in S+ARRAYANALYZER, so only the first two non-empty lines in this block are used.

Each line that defines a factor has a specific format:

[factor code] | [factor name] | [levels]

where **[factor code]** is A, B, etc. **[factor name]** is the name you wish to use for the factor, and **[levels]** is a space or comma delimited list of level names to use for each factor. Each element in the line is separated by a vertical bar character "|" (also known as the "pipe" symbol).

As an example, consider the line:

A|CondTime|Swim3wks Swim4wks+1wk NoSwim4wks NoSwim4wks+1wk

The factor code is **A**. The factor name is **CondTime**. The levels for this factor are **Swim3wks**, **Swim4wks+1wk**, **NoSwim4wks**, and **NoSwim4wks+1wk**.

Design valuenames

The lines in a design block are different than other data blocks. There is only one valuename allowable in this block (listed in a table below). The other non-empty lines in this block are interpreted as files to import and factors and levels to use for each of these files.

Each filename line is comma delimited and has a specific format:

[file] [name], [factor A level], [factor B level], ...

where **[file]** is either the full pathname to the file to import or is just the filename without path. If that is the case, all the other files in this block must be from the same path and the valuename **RelativePath** should be defined (see table below). If **RelativePath** is not defined, then the current working directory is used to find the files.

[name] is the name to assign this array to import.

[factor A level] is the level to use for this file for factor A. The level specified here should match one of the levels for this factor defined in the **FactorInfo** block (see above), and likewise for **[factor B level]**, etc.

Currently only two factors are allowed in S+ARRAYANALYZER, so only the third and fourth delimited values in the lines are used to define levels for each factor. As an example, consider the lines:

;FactorInfo[START]

A|CondTime|Swim3wks Swim4wks Swim4wks+1wk NoSwim4wks NoSwim4wks+1wk

;FactorInfo[END]

;Design[START]

Swim3w1.txt,Swim3wks1,Swim3wks

;Design[END]

The file to import is **Swim3w1.txt**. Since there is no path specified to the file here and since the valuename **RelativePath** was not specified in the block, the current working directory is used to find **Swim3w1.txt**. The name is **Swim3wks1**. The factor A level for this file is Swim3wks. Notice that **Swim3wks** appears as an allowable level for factor A in the **FactorInfo** block.

Table J.5: *Table of* ***Design*** *valuenames.*

Value Name	Rules	Example
RelativePath	OPTIONAL If not specified, and no full path is specified for a file, the current working directory will be used. Specifies the path to use for all files in the block.	;Design[START] relativepath=C:\Program Files\Insightful\splus62\module\ArrayAnalyzer\examples swirl.1.spot,swirl.1,Swirl,WidType swirl.2.spot,swirl.2,Swirl,WidType swirl.3.spot,swirl.3,WidType,Swirl swirl.4.spot,swirl.4,WidType,Swirl ;Design[END]

DesignType valuenames

Table J.6: *Table of **DesignType** valuenames.*

Value Name	Rules	Example
Type	REQUIRED. Defines the type of design used. Can be one of the following values only: 0 1 2 3 where 0 = two sample 1 = loop 2 = reference 3 = other	Type=0
TWO_SAMPLE_DyeSwap	OPTIONAL. Only used if Type == 0 (two sample). Determines whether dye swapping is done. True is indicated by any of the following values: Y Yes T True 1 All other values indicate false.	TWO_SAMPLE_DyeSwap

Table J.6: *Table of* ***DesignType*** *valuenames. (Continued)*

Value Name	Rules	Example
LOOP_Factor	OPTIONAL. Only used if Type == 1 (loop). Specifies the factor to use. This factor must be a valid factor name from the FactorInfo block.	LOOP_Factor=ZebraFish
LOOP_DyeSwap	OPTIONAL. Only used if Type == 1 (loop). Determines whether dye swapping is done. True is indicated by any of the following values: Y Yes T True 1 All other values indicate false.	LOOP_DyeSwap=1
REFERENCE_Factor	OPTIONAL. Only used if Type == 2 (reference). Specifies the factor to use. This factor must be a valid factor name from the FactorInfo block.	REFERENCE_Factor= ZebraFish

Table J.6: *Table of* ***DesignType*** *valuenames. (Continued)*

Value Name	Rules	Example
REFERENCE_RefLevel	OPTIONAL. Only used if Type == 2 (reference). Specifies the factor level to use. This level must be a valid factor level name from the FactorInfo block for the factor specified in REFERENCE_Factor.	REFERENCE_RefLevel= Swirl
REFERENCE_RefType	OPTIONAL. Only used if Type == 2 (reference). Specifies the reference type. Allowable values are: 0 1 2 where 0 = REF_CY3 1 = REF_CY5 2 = REF_DYESWAP	REFERENCE_RefType=2

Example Design File for Affymetrix MAS/CEL

The following is **swimming.txt**, the design file used in the section One-Way Design in Chapter 2, Examples: Affymetrix MAS Data. This file can be found in the **examples** directory of S+ARRAYANALYZER. This experiment has two factor levels and 6 replicates of each level. As an example, let's only read four of the **0week** chips and all six of the **4week** chips. The two lines with blank file names are required to give six replicates for each level:

```
;V1.3
;ImportInfo[START]
FileType=MAS 5 Summary Data
```

```
ChipName=mgu74av2
CDFPath=
SaveAs=myExprSet
PrintOutput=1
;ImportInfo[END]
;FactorInfo[START]
A|CondTime|Swim3wks Swim4wks Swim4wks+1wk NoSwim4wks
NoSwim4wks+1wk
;FactorInfo[END]
;Design[START]
relativepath=C:\Program
Files\Insightful\splus62\module\ArrayAnalyzer\examples
Swim3w1.txt,Swim3wks1,Swim3wks
Swim3w2.txt,Swim3wks2,Swim3wks
Swim3w3.txt,Swim3wks3,Swim3wks
Swim4w1.txt,Swim4wks1,Swim4wks
Swim4w2.txt,Swim4wks2,Swim4wks
Swim4w3.txt,Swim4wks3,Swim4wks
Swim4w1w1.txt,Swim4wks+1wk1,Swim4wks+1wk
Swim4w1w2.txt,Swim4wks+1wk2,Swim4wks+1wk
Swim4w1w3.txt,Swim4wks+1wk3,Swim4wks+1wk
NoSwim4w1.txt,NoSwim4wks1,NoSwim4wks
NoSwim4w2.txt,NoSwim4wks2,NoSwim4wks
NoSwim4w3.txt,NoSwim4wks3,NoSwim4wks
NoSwim4w1w1.txt,NoSwim4wks+1wk1,NoSwim4wks+1wk
NoSwim4w1w2.txt,NoSwim4wks+1wk2,NoSwim4wks+1wk
NoSwim4w1w3.txt,NoSwim4wks+1wk3,NoSwim4wks+1wk
;Design[END]
```

Example Design File for Two-Color Experiments

The following is **designSwirl.txt**, the design file used in the section Two-Sample Design in Chapter 4, Examples: Two-Color Data.

The design format is quite similar to that used for importing Affymetrix MAS or CEL data.

```
;V1.3
;ImportInfo[START]
ChipName=SwirlLayout
```

```
SaveAs=SwirlMarrayRaw
PrintOutput=1
;ImportInfo[END]
;FactorInfo[START]
A|ZebraFish|No|ZebraFish1 ZebraFish2
;FactorInfo[END]
;DesignType[START]
Type=0
TWO_SAMPLE_DyeSwap=1
LOOP_Factor=ZebraFish
LOOP_DyeSwap=1
REFERENCE_Factor=ZebraFish
REFERENCE_RefLevel=ZebraFish1
REFERENCE_RefType=2
;DesignType[END]
;Design[START]
relativepath=C:\program
files\insightful\splus62\module\arrayanalyzer\examples
swirl.1.spot,swirl.1,ZebraFish1,ZebraFish2
swirl.2.spot,swirl.2,ZebraFish1,ZebraFish2
swirl.3.spot,swirl.3,ZebraFish2,ZebraFish1
swirl.4.spot,swirl.4,ZebraFish2,ZebraFish1
;Design[END]
```

APPENDIX B: IMPORTING DATA

Introduction **386**
Supported Affymetrix MAS/CEL Data Formats 386
ASCII Data 387
Excel Data 387
Layout Information for Two-Channel Data 388

INTRODUCTION

S+ARRAYANALYZER can import many different microarray data file formats. For example, Affymetrix data files are available in several forms as shown in the table below. The special binary formats are auto-detected by the import dialogs. Two-channel (cDNA) data is typically in ASCII or Excel files.

This appendix provides guidelines for importing data successfully into S+ARRAYANALYZER.

Supported Affymetrix MAS/CEL Data Formats

The table below lists the various Affymetrix MAS/CEL data formats supported by S+ARRAYANALYZER 2.0.

Table B.5: *Supported Affymetrix MAS/CEL data formats.*

Format	Type	Extension	Notes
MAS 4	ASCII	.txt, .csv	Older Affymetrix format, infrequently used.
MAS 5	ASCII	.txt, .csv	Standard summarized Affymetrix data.
MAS 5	Excel	.xls	Excel version of MAS 5 file.
CHP	binary	.CHP	Binary version of summarized data.
CEL	ASCII	.CEL	Raw CEL data.
CEL	binary	.CEL	Binary raw CEL data.

Note that compressed binary files are not supported in this version of S+ARRAYANALYZER.

ASCII Data

For either Affymetrix Summary (MAS) or two-channel ASCII data, the ASCII files must meet these requirements:

1. The data for each array is in a separate file. For two-channel data, data for both channels must be in the same file.
2. The data must be delimited by a single common delimiter, typically a tab or comma.
3. The data can have a header, the end of the header section must be a line of columns names for the data. The column names must be unique. If there is the no header then the first line is the column names row.
4. From the column names row to the end of the file, each line (row) must have the same number of delimiters.
5. Each file must have the same number of lines. If there is a header section, each file must have the same number of lines in the header.
6. A tail section is not allowed.

The import dialogs will attempt to auto-detect the number of lines in the header and the delimiter used in the file. In some case this detection may fail. The user can explicitly set the number of lines to skip and the delimiter on the Options tab.

Note that opening a delimited ASCII file in Excel and then saving it as a delimited ASCII file can result in a file that violates rule 4 above if there are blank cells at the end of the lines. The first few lines will include the delimiters at the end of the lines but then Excel will stop outputting the delimiters in subsequent lines.

Excel Data

For either Affymetrix Summary (MAS) or two-channel Excel data, the files must meet these requirements:

1. The data for each array is in a separate file. For Two-Channel data, data for both channels must be in the same file.
2. The first line of the file must a column name row with unique values.
3. There cannot be any tail section.

Layout Information for Two-Channel Data

To read two-channel data, information about the layout must be read first. This includes information about the spot grid rows and columns, gene names, and control spots. Sometimes this information is included in the data file, and other times it is available in a separate file (e.g., a GAL file for GenePix data). The GAL file versions are currently all 1.0, even if they are created by different GenePix versions.

In S+ARRAYANALYZER 2.0, the layout information must be in an ASCII file; it cannot be read from an Excel file. If you have your layout information in an Excel file, you can export it as a delimited ASCII file from Excel.

INDEX

A

annotate library 341

B

BH 144
Bonferroni 84, 88, 144, 145, 188
Box Plot 140

C

cDNA normalization
    median 140, 183
    printTipMAD 140
    scalePrintTipMAD 142
Chromosome Plot 90
Chromosome plot 84

F

FDR 84, 143
FWER 84, 143, 144

G

GenBank 88, 145

H

Heat Map 89, 146
Heat map 84
heat map 92, 144, 321, 324, 340

I

INSTALL.txt 6
installing I-Miner 6

L

LocusLink 9, 92, 145, 340, 341, 360

M

medianIQR 338, 341
MIAME 75, 100, 129
MvA plot 141

P

paired-t 142, 144
platforms, supported 6

Q

QQ Norm Plot 147

R

requirements, system 6

S

support, technical 7
supported platforms 6
system requirements 6

T

technical support 7

V

Variance Plot 91
Variance plot 85
Volcano Plot 145
Volcano plot 84, 189
volcano plot 2, 88, 339

W

Wilcoxon test 142, 144